LENORA COLLEGE OF ENGINEERING

Rampachodavaram
E.G. Dist., A.P.

CERTIFICATE

Roll No.

This is to Certify that _____

Student of B. Tech _____ in Lenora College of Engineering. Rampachodavaram, E.G. Dt., He/She has Successfully complete His/ Her Record work in _____

during the academic year 20 - 20

Examine :

Internal :

Externam :

Lab Incharge :

Head of the Department

LENORA COLLEGE OF ENGINEERING

RAMPACHODAVARAM

R19 – M.Tech CSE

1st Year 2nd Semester

MEAN STACK TECHNOLOGIES LAB

Experiment 1:

Develop static pages (using only HTML) of an online Book store. The pages should resemble www.amazon.com. The website should consist of the following pages:
- Home page
- Registration and user Login
- User profile page
- Books catalog
- Shopping cart
- Payment by credit card Order Confirmation

Home page

Main.html:

```
<html>
<head>
<title>
Amazon</title>
</head>
<body bgcolor="cyan"> <center>
<strong><h1>Welcome to AMAZON</h1></strong>
<form method="post" action="login.html" target=_blank >
<h4>for books</h4><input type="submit" value="click here">
</form>
</center>
</body>
</html>
```

Registration and user Login

Login.html:

```
<html>
<head>
<title>
login</title>
</head>
<body bgcolor="cyan"> <center>
<strong><h1> AMAZON </h1></strong></center>
<right>
<table align="right">
<tr>
<td><h4>user name</td>
<td><input type="text" ></td>
<td></td>
</tr>
<tr>
<td><h4>password</td>
```

```
<td><input type="password"></td>
<td></td>
</tr>
<tr>
<td>
<form method="post" action="catalog.html" >
<input type="submit" value="submit" >
</form>
</td>
<td>
<form method="post" action="reg.html" >
<input type="submit" value="register" >
  
<input type="reset" value="reset"></form></td>
</tr>
</table>
</body>
</html>
```

Registration page

reg.html:

```
<html>
<head>
<title>
login page</title>
</head>
<body bgcolor="cyan">
<center><strong><h1> AMAZON </h1></strong></center>
<form method="post" action="catalog.html" >
<right>
<table align="left">
<tr>
<td><h4>user name</td>
<td><input type="text" ></td>
<tr>
<tr>
<td><h4>password</td>
<td><input type="password"></td>
</tr>
<tr>
<td><h4>confirm password</td>
<td><input type="password"></td>
</tr>
<tr>
<td><h4>male   
<option >
<input type="radio" name="sex" id="male"></td>
<td><h4>female    
<input type="radio" name="sex" id="female" ></td>
</option>
```

```
</tr>
<tr>
<td>Address</td>
<td><textarea name="address" rows=5 cols=19>
</textarea>
</td>
<tr>
<td>
<input type="submit" value="submit" ></td>
<td>
<input type="reset" value="reset"></td>
</tr>
</form>
</body>
</html>
```

Userprofile

userprofile.html

```
<html>
<head>
<title>
userprofile</title>
</head>
<body bgcolor="cyan"> <center>
<strong><h1>Welcome to AMAZON Online Book Store </h1></strong></center>
Edit your profile here...
<form method="post" action="catalog.html" >
<right>
<table align="left">
<tr>
<td><h4>Edit user name</td>
<td><input type="text" ></td>
<tr>
<tr>
<td><h4>Edit password</td>
<td><input type="password"></td>
</tr>
<tr>
<option >
<td><h4>male   
<input type="radio" name="sex" id="male"></td>
<td><h4>female    
<input type="radio" name="sex" id="female" ></td>
</option>
</tr>
<tr>
<td>Edit Address</td>
<td><textarea name="address" rows=5 cols=19>
</textarea>
</td>
```

```
<tr>
<td>
<input type="submit" value="submit" ></td>
</tr>
</form>
</body>
</html>
```

Books catalog
Catalog.html:

```
<html>
<head>
<title>
books catalog</title>
</head>
<body bgcolor="cyan">
<center><h1>AMAZON</h1></center>
<form method="post" action="shopping.html">
<left>
<table>
<tr>
<td><b><h3>frontend books</td>
<td></td></tr>
<tr>
<td></td>
<td><h4>C&Ds</td>
</tr>
<tr>
<td></td>
<td><h4>Ads</td>
</tr>
<tr>
<td></td>
<td><h4>JAVA
</td></tr>
<tr>
<td><b><h3>backend books</td>
<td></td>
</tr>
<tr>
<td></td>
<td><h4>Oracle</td>
</tr>
<tr>
<td></td>
<td><h4>Ms SQL Server
</td></tr>
<tr>
<td></td>
<td><h4>MySql </td>
```

```
</tr>
</table>
</h4>
<center>
<b>for buy one of these books
<br>
</b><input type="submit" value="click here">
</center>
</form>
</body>
</html>
```

Shopping cart

Shopping.html:

```
<html>
<head><title>shopping cart</title>
</head>
<body bgcolor="cyan">
<center><h1>
Shopping Cart</h1></center>
<br><br><br><br><br>
<table align="center">
<tr>
<td>Text Books</td>
<td>
<select >
<optgroup label="select the book">
<option value="C&Ds">C&Ds
<option value="Ads">Ads
<option value="Java">Java
<option value="Oracle">Oracle
<option value="Ms SQL Server">Ms SQL Server
<option value="MySql">MySql
</optgroup>
</select>
</td></tr>
<tr>
<td>
Quantity</td>
<td>
<input type="text" id="q">
</td></tr>
<tr>
<td></td>
<td>
<form method=post action="payment.html">
<input type="submit" value=ok />
</form>
</td></tr>
</table>
```

```
<center>
<pre>Cost of one book is"500" + shipping "100"</pre>
</center>
<body>
</html>
```

Payment by credit card

Payment.html:

```
<html>
<head><title>payment</title></head>
<body bgcolor="cyan">
<center><h1>Payment By Credit Card</h1></center>
<form method=post action="ordrconform.html">
<br><br><br><br><br>
<table align="center">
<tr>
<td>
<h4>Total Amount</h4></td>
<td><input type="text">
</td>
</tr>
<tr>
<td><h4>Credit Card Number</td>
<td><input type="text"></td>
</tr>
<tr>
<td>
</td>
<td><input type="submit" value=OK>
</td>
</tr>
</table>
</form></body>
</html>
```

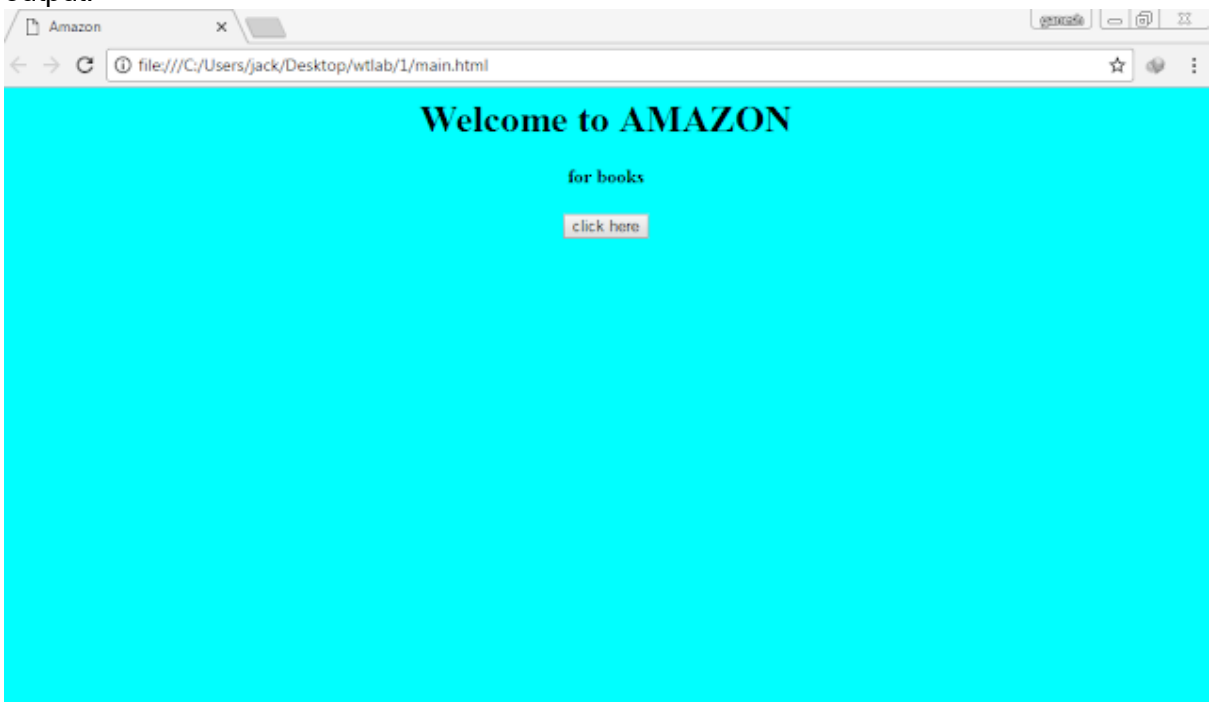
Order Conformation

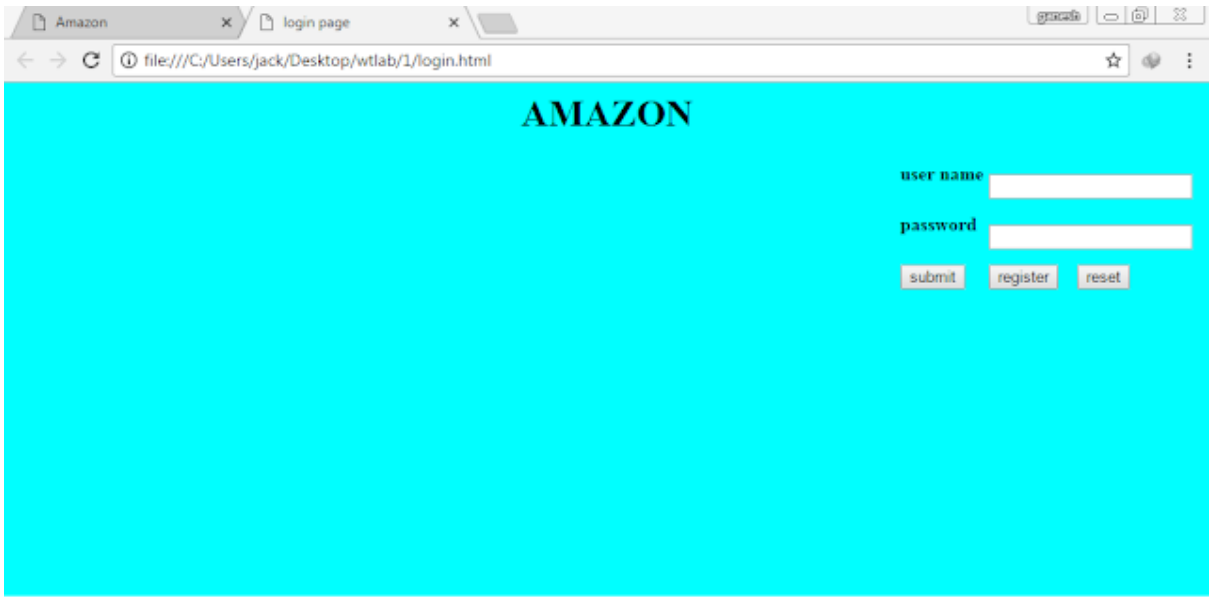
Ordrconform:

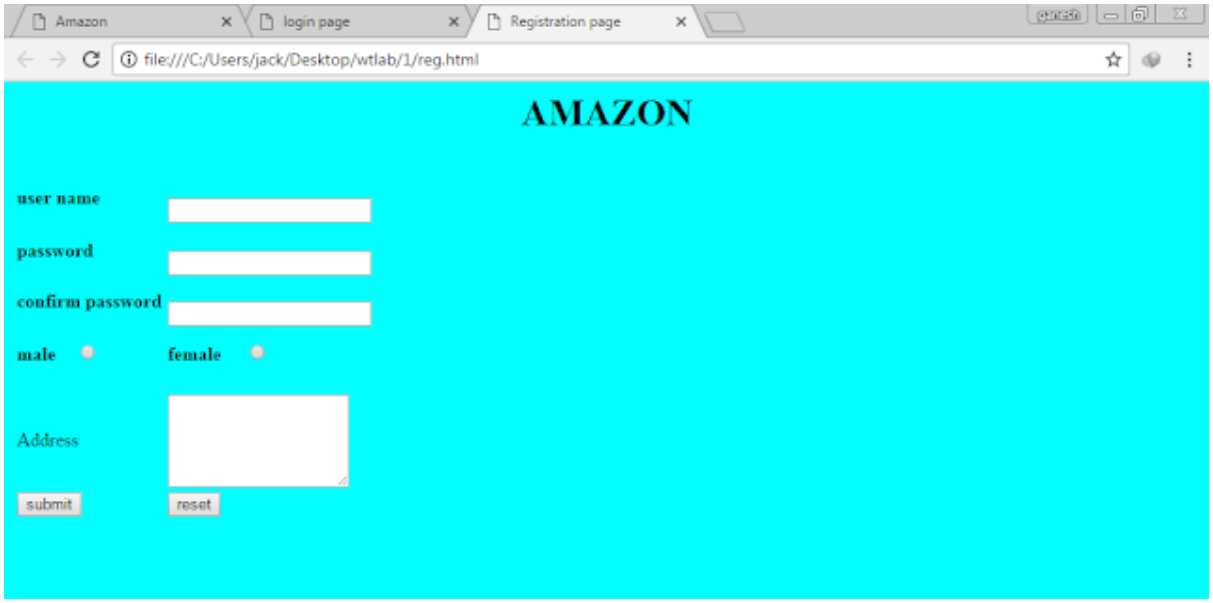
```
<html>
<head><title>order conformation</title><M/head>
<body bgcolor="cyan">
<center>
<h1><b>BOOK SHOPPING</h1>
<pre><strong>
<b>Your order Is Conformed
</strong></pre>
<h2><b>THANK YOU</h2>
</center>
```

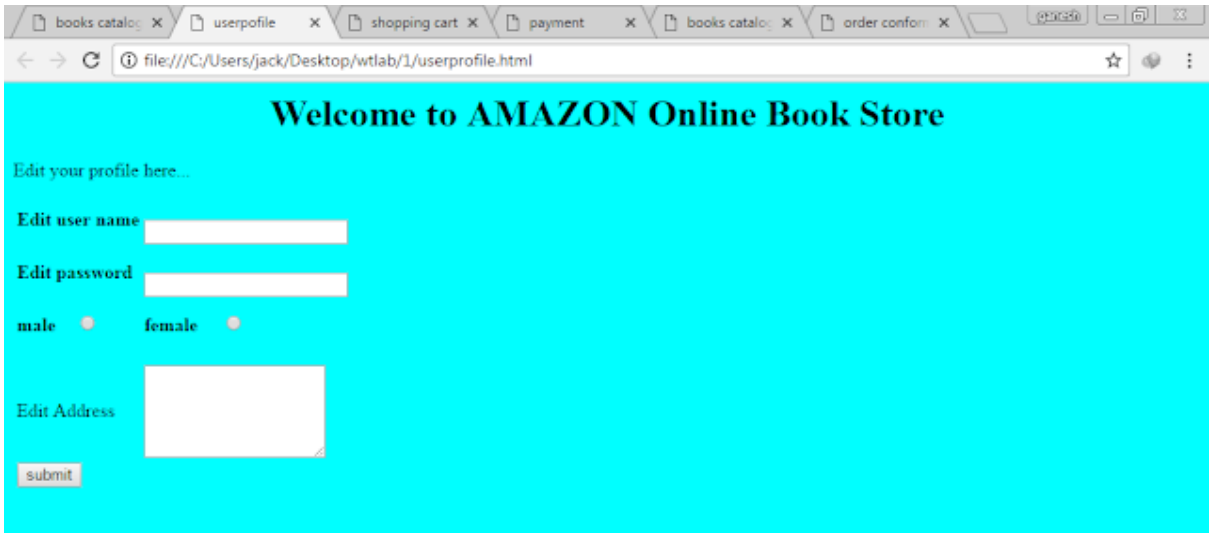
</body></html>

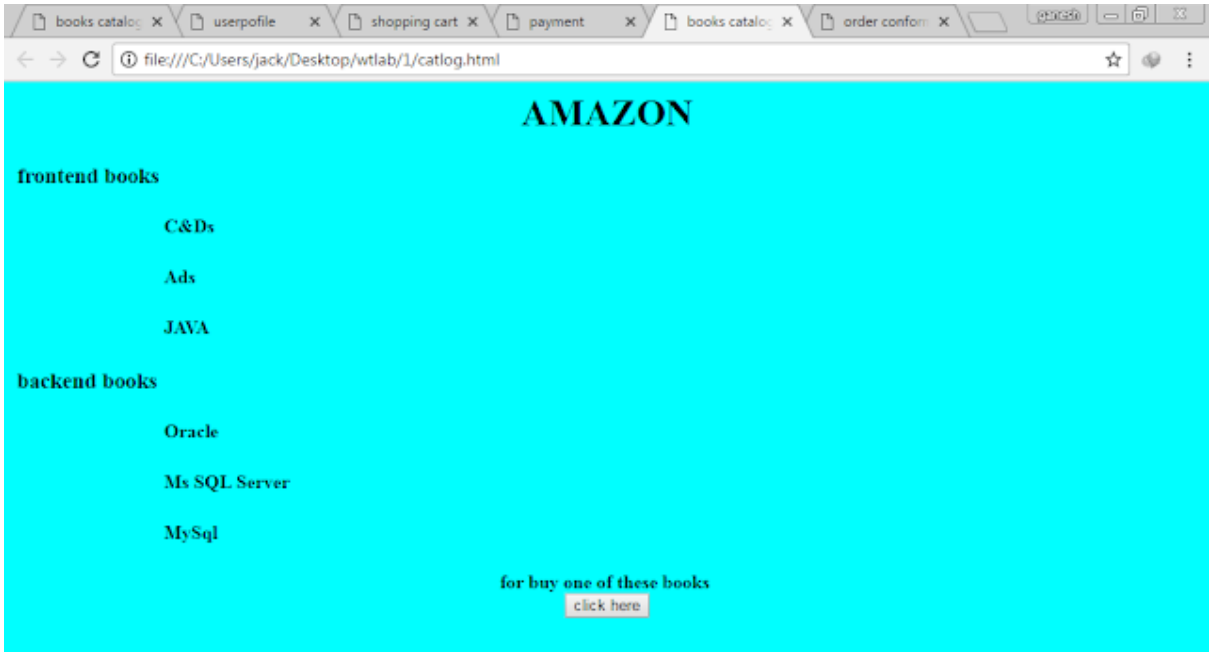
output:

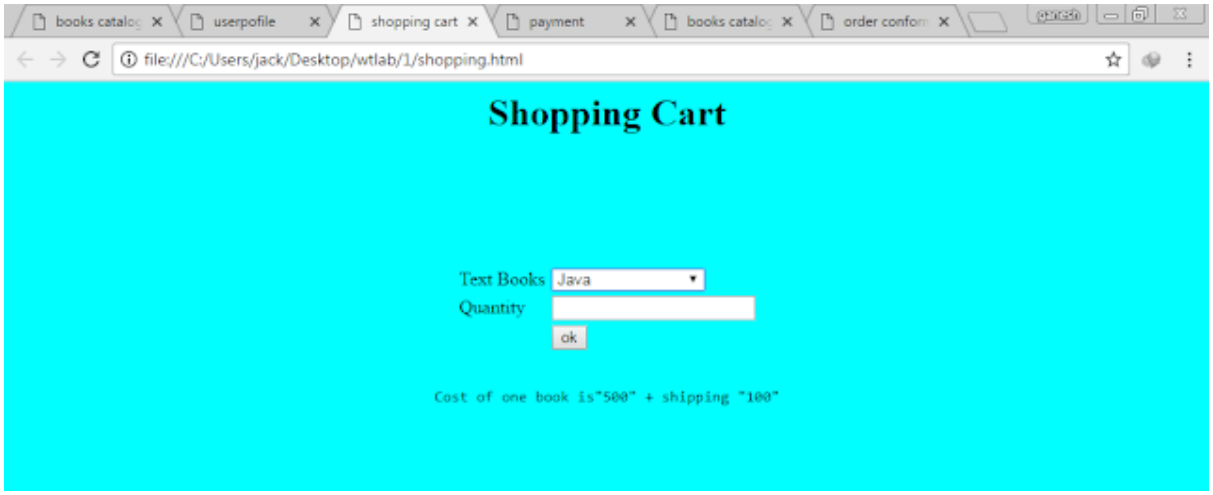




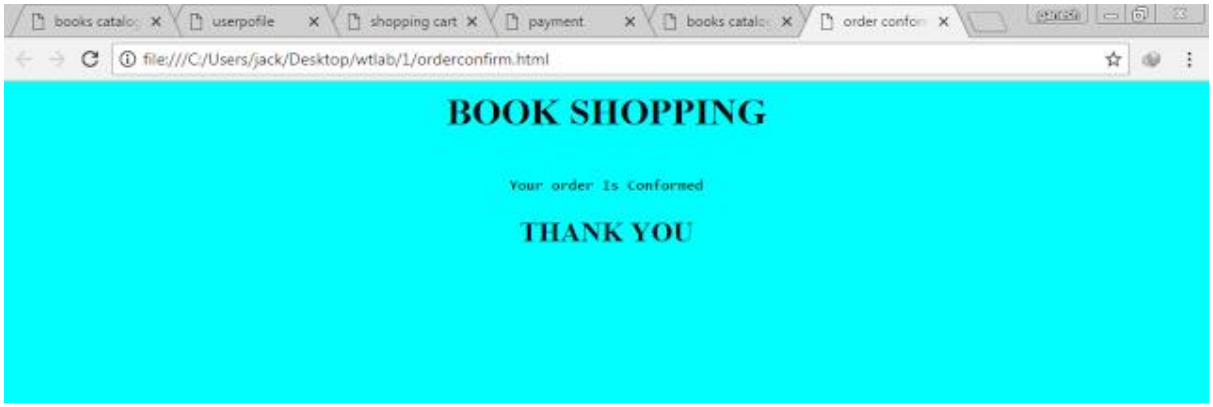

Amazon
login page
Registration page
file:///C:/Users/jack/Desktop/wtlab/1/reg.html
AMAZON
user name
password
confirm password
male
female
Address
submit
reset


books catalog
userpofile
shopping cart
payment
books catalog
order confirm
file:///C:/Users/jack/Desktop/wtlab/1/userprofile.html
Welcome to AMAZON Online Book Store
Edit your profile here...
Edit user name
Edit password
male
female
Edit Address
submit


books catalog
userpofile
shopping cart
payment
books catalog
order confirm
file:///C:/Users/jack/Desktop/wtlab/1/catlog.html
AMAZON
frontend books
C&Ds
Ads
JAVA
backend books
Oracle
Ms SQL Server
MySql
for buy one of these books
click here


books catalog
userpofile
shopping cart
payment
books catalog
order confirm
file:///C:/Users/jack/Desktop/wtlab/1/shopping.html
Shopping Cart
Text Books
Java
Quantity
ok
Cost of one book is"500" + shipping "100"

books catalog
userpofile
shopping cart
payment
books catalog
order confirm
file:///C:/Users/jack/Desktop/wtlab/1/payment.html
Payment By Credit Card
Total Amount
Credit Card Number
OK


books catalo
userpofile
shopping cart
payment
books catalo
order confor
file:///C:/Users/jack/Desktop/wtlab/1/orderconfirm.html
BOOK SHOPPING
Your order Is Conformed
THANK YOU

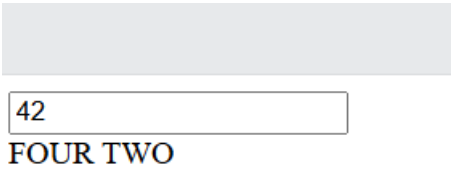
Experiment-2:
Write an HTML page including any required JavaScript that takes a number from text field in the range of 0 to 999 and shows it in words. It should not accept four and above digits, alphabets and special characters.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <input type="number" onchange="f1()"/>
    <div id="ans"></div>
    <script>
        function f1()
        {
            var v=event.target.value;
            if(v>999 || v<0)
            {
                alert("Invalid number(range is 0-999)");
            }
            else
            {
                var ans="";
                //const strRev =  [...v].reverse().join("");
                const strRev=v;
                for(var i=0;i<strRev.length;i++)
                {
                    var d=strRev.charAt(i);
                    //alert(d);
                    switch(d)
```

```
{
    case "1":
        ans+="ONE ";
        break;

    case "2":
        ans+="TWO ";
        break;

    case "3":
        ans+="THREE ";
        break;

    case "4":
        ans="FOUR ";
        break;

    case "5":
        ans="Five ";
        break;

    case "6":
        ans="SIX ";
        break;

    case "7":
        ans="SEVEN ";
        break;

    case "8":
        ans="EIGHT ";
        break;
```

```
                case "9":
                    ans="NINE ";
                    break;

                case "0":
                    ans="ZERO ";
                    break;
                }
            }
            document.getElementById("ans").innerHTML=ans;
        }
    }
  </script>
</body>
</html>
```

OUTPUT:



Experiment-3:
Develop and demonstrate JavaScript with POP-UP boxes and functions for the following problems:
a) Display Date using onClick function.
b) Factorial of a number using alert.
c) Multiplication table of a number using alert.
d) Sum of entered numbers using alert and confirm.

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript Pop-up Box Examples</title>
<script>
// a) Display Date using onClick function
function displayDate() {
    document.getElementById("dateOutput").innerHTML = new Date();
}

// b) Factorial of a number using alert
function calculateFactorial() {
    let num = prompt("Enter a number to calculate its factorial:");
    num = parseInt(num);

    if (isNaN(num) || num < 0) {
        alert("Please enter a non-negative integer.");
        return;
    }

    let factorial = 1;
    for (let i = 1; i <= num; i++) {
        factorial *= i;
    }
    alert("The factorial of " + num + " is: " + factorial);
}
```

```
// c) Multiplication table of a number using alert
function displayMultiplicationTable() {
    let num = prompt("Enter a number to display its multiplication table:");
    num = parseInt(num);

    if (isNaN(num)) {
        alert("Please enter a valid number.");
        return;
    }

    let table = "Multiplication Table for " + num + ":\n";
    for (let i = 1; i <= 10; i++) {
        table += num + " x " + i + " = " + (num * i) + "\n";
    }
    alert(table);
}

// d) Sum of entered numbers using alert and confirm
function calculateSum() {
    let sum = 0;
    let continueAdding = true;

    while (continueAdding) {
        let numStr = prompt("Enter a number to add:");
        let num = parseFloat(numStr);

        if (isNaN(num)) {
            alert("Invalid input. Please enter a number.");
        } else {
            sum += num;
        }
```

```
        continueAdding = confirm("Do you want to add another number?");
    }
    alert("The total sum of entered numbers is: " + sum);
}
</script>
</head>
<body>

<h1>JavaScript Pop-up Box Demonstrations</h1>

<h2>a) Display Date</h2>
<button onclick="displayDate()">Display Current Date</button>
<p id="dateOutput"></p>

<h2>b) Factorial of a Number</h2>
<button onclick="calculateFactorial()">Calculate Factorial</button>

<h2>c) Multiplication Table</h2>
<button onclick="displayMultiplicationTable()">Show Multiplication Table</button>

<h2>d) Sum of Entered Numbers</h2>
<button onclick="calculateSum()">Calculate Sum</button>

</body>
</html>
```

Output:

JavaScript Pop-up Box Demonstrations

a) Display Date

Display Current Date

b) Factorial of a Number

Calculate Factorial

c) Multiplication Table

Show Multiplication Table

d) Sum of Entered Numbers

Calculate Sum

Experiment-4:
Create a simple visual bean with an area filled with a color. If shape is true, it's Square, else Circle. Color should change dynamically on mouse click.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dynamic Bean</title>
    <style>
        #bean-container {
            width: 100px;
            height: 100px;
            margin: 50px auto;
            cursor: pointer;
            border: 1px solid #ccc; /* Optional: for visibility */
        }
        .square {
            border-radius: 0;
        }
        .circle {
            border-radius: 50%;
        }
    </style>
</head>
<body>

    <div id="bean-container" class="square"></div>

    <script>
        const beanContainer = document.getElementById('bean-container');
        let isSquare = true; // Initial shape
        const colors = ['red', 'blue', 'green', 'purple', 'orange'];
```

```
        let currentColorIndex = 0;

        // Set initial color
        beanContainer.style.backgroundColor = colors[currentColorIndex];

        beanContainer.addEventListener('click', () => {
            // Change color
            currentColorIndex = (currentColorIndex + 1) % colors.length;
            beanContainer.style.backgroundColor = colors[currentColorIndex];

            // Toggle shape
            isSquare = !isSquare;
            if (isSquare) {
                beanContainer.classList.remove('circle');
                beanContainer.classList.add('square');
            } else {
                beanContainer.classList.remove('square');
                beanContainer.classList.add('circle');
            }
        });
    </script>

</body>
</html>
```

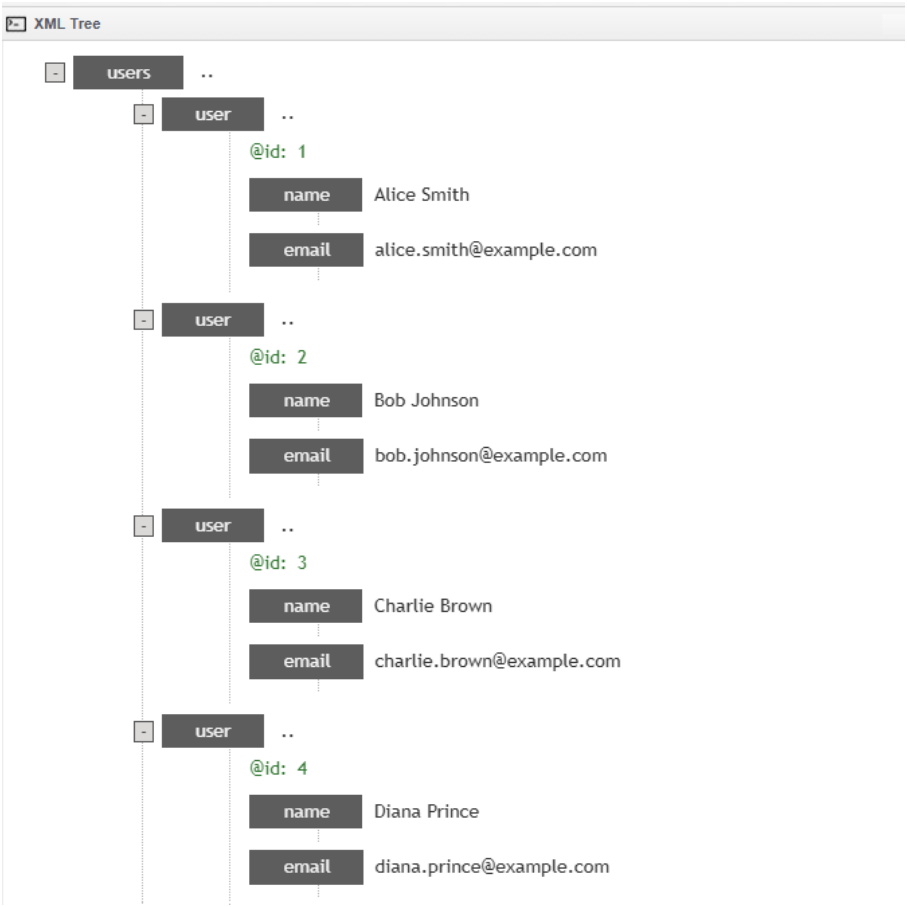
Output:

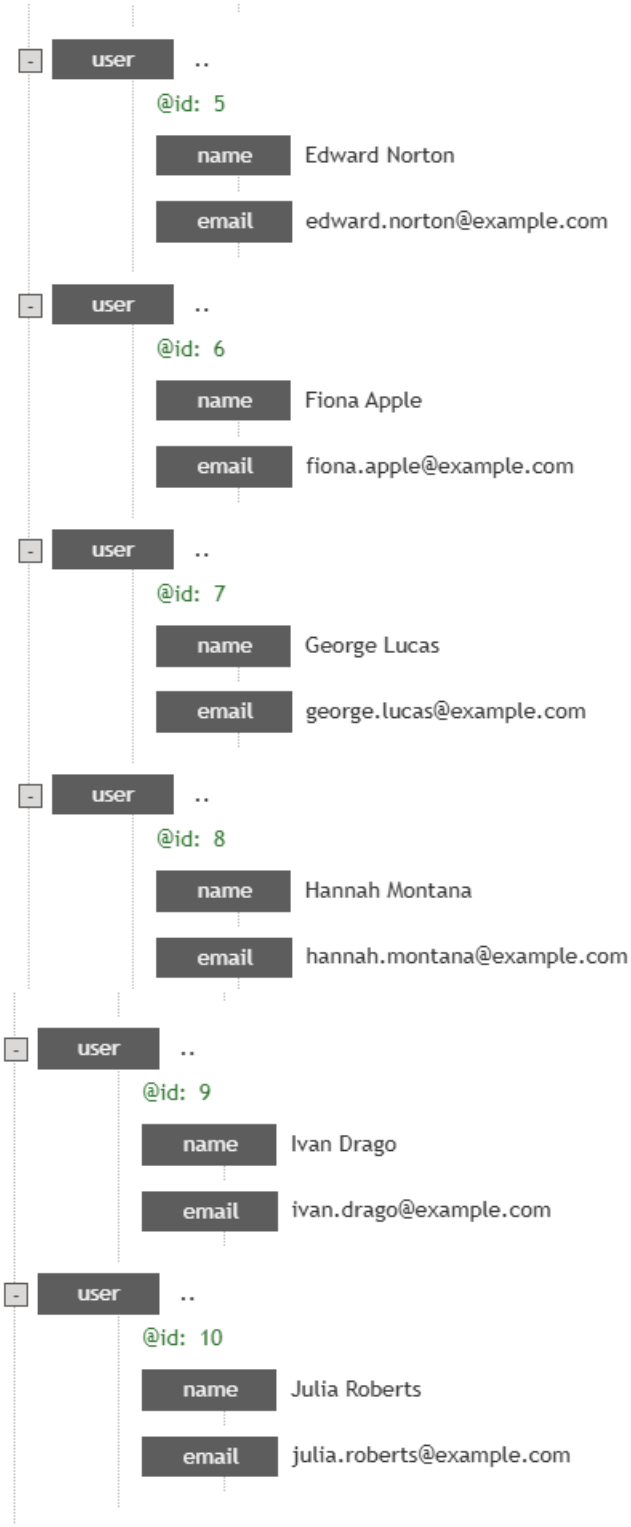
Experiment-5:
Create an XML document with 10 users. Write a Java program to fetch user details using DOM or SAX parser based on input user ID.

```
<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user id="1">
        <name>Alice Smith</name>
        <email>alice.smith@example.com</email>
    </user>
    <user id="2">
        <name>Bob Johnson</name>
        <email>bob.johnson@example.com</email>
    </user>
    <user id="3">
        <name>Charlie Brown</name>
        <email>charlie.brown@example.com</email>
    </user>
    <user id="4">
        <name>Diana Prince</name>
        <email>diana.prince@example.com</email>
    </user>
    <user id="5">
        <name>Edward Norton</name>
        <email>edward.norton@example.com</email>
    </user>
    <user id="6">
        <name>Fiona Apple</name>
        <email>fiona.apple@example.com</email>
    </user>
    <user id="7">
        <name>George Lucas</name>
        <email>george.lucas@example.com</email>
    </user>
```

```
    <user id="8">
        <name>Hannah Montana</name>
        <email>hannah.montana@example.com</email>
    </user>
    <user id="9">
        <name>Ivan Drago</name>
        <email>ivan.drago@example.com</email>
    </user>
    <user id="10">
        <name>Julia Roberts</name>
        <email>julia.roberts@example.com</email>
    </user>
</users>
```

Output:


XML Tree
users
user
@id: 1
name
Alice Smith
email
alice.smith@example.com
user
@id: 2
name
Bob Johnson
email
bob.johnson@example.com
user
@id: 3
name
Charlie Brown
email
charlie.brown@example.com
user
@id: 4
name
Diana Prince
email
diana.prince@example.com


user ..
@id: 5
name Edward Norton
email edward.norton@example.com
user ..
@id: 6
name Fiona Apple
email fiona.apple@example.com
user ..
@id: 7
name George Lucas
email george.lucas@example.com
user ..
@id: 8
name Hannah Montana
email hannah.montana@example.com
user ..
@id: 9
name Ivan Drago
email ivan.drago@example.com
user ..
@id: 10
name Julia Roberts
email julia.roberts@example.com

Experiment-6:
Develop and demonstrate PHP scripts:
a) Sum of individual digits
b) Check if a number is palindrome

a) Sum of individual digits

```
<?php

function sumDigits(int $number): int {
    $sum = 0;
    // Handle negative numbers by taking the absolute value
    $number = abs($number);

    // Loop while the number is greater than 0
    while ($number > 0) {
        // Get the last digit using the modulo operator
        $digit = $number % 10;
        // Add the digit to the sum
        $sum += $digit;
        // Remove the last digit by integer division
        $number = (int)($number / 10);
    }
    return $sum;
}

// Demonstration
$num1 = 12345;
echo "Sum of digits of " . $num1 . ": " . sumDigits($num1) . "\n"; // Output: 15

$num2 = 789;
echo "Sum of digits of " . $num2 . ": " . sumDigits($num2) . "\n"; // Output: 24
```

```
$num3 = 0;
echo "Sum of digits of " . $num3 . ": " . sumDigits($num3) . "\n"; // Output: 0

$num4 = -567;
echo "Sum of digits of " . $num4 . ": " . sumDigits($num4) . "\n"; // Output: 18

?>
```

Output:

```
Output
Sum of digits of 12345: 15
Sum of digits of 789: 24
Sum of digits of 0: 0
Sum of digits of -567: 18
```

b) Check if a number is palindrome

```
<?php

function isPalindrome(int $number): bool {
    // Handle negative numbers as non-palindromes (or adjust logic as needed)
    if ($number < 0) {
        return false;
    }

    $originalNumber = $number;
    $reversedNumber = 0;

    // Reverse the number
    while ($number > 0) {
        $digit = $number % 10;
        $reversedNumber = ($reversedNumber * 10) + $digit;
```

```
        $number = (int)($number / 10);
    }

    // Compare the original and reversed numbers
    return $originalNumber === $reversedNumber;
}

// Demonstration
$pal1 = 121;
echo $pal1 . " is a palindrome: " . (isPalindrome($pal1) ? "True" : "False") . "\n"; // Output: True

$pal2 = 12345;
echo $pal2 . " is a palindrome: " . (isPalindrome($pal2) ? "True" : "False") . "\n"; // Output: False

$pal3 = 9;
echo $pal3 . " is a palindrome: " . (isPalindrome($pal3) ? "True" : "False") . "\n"; // Output: True

$pal4 = 1221;
echo $pal4 . " is a palindrome: " . (isPalindrome($pal4) ? "True" : "False") . "\n"; // Output: True

$pal5 = -121;
echo $pal5 . " is a palindrome: " . (isPalindrome($pal5) ? "True" : "False") . "\n"; // Output: False

?>
```

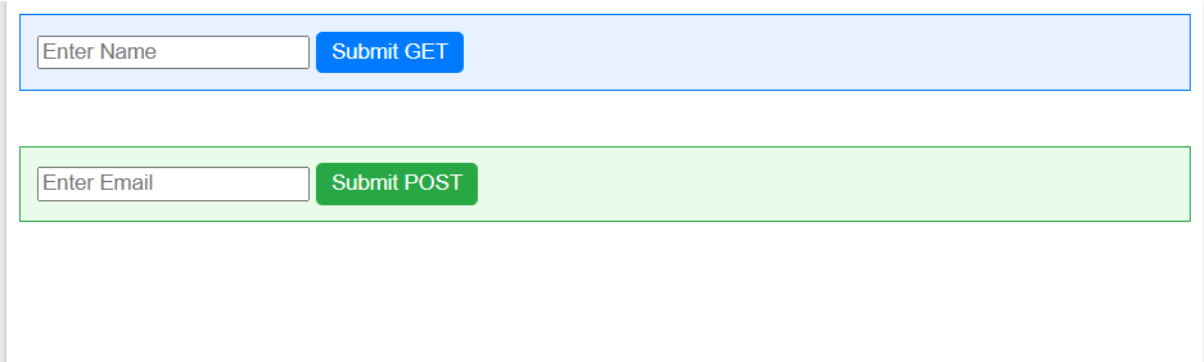
Output:

```
Output
121 is a palindrome: True
12345 is a palindrome: False
9 is a palindrome: True
1221 is a palindrome: True
-121 is a palindrome: False
```

Experiment-7:
Implement in CSS:
a) 'get' and 'post' methods
b) Colors, border padding
c) Buttons, frames, tables, nav bars

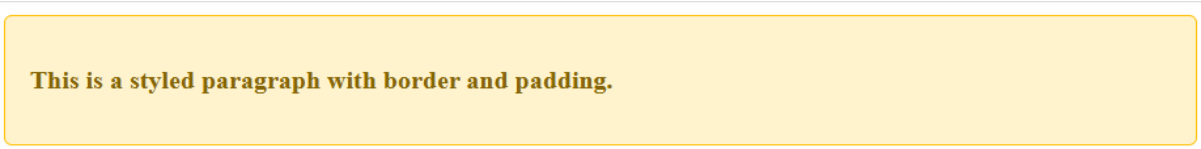
a) 'get' and 'post' methods

```
<!-- GET method form -->

<form method="get" style="background-color:#e7f1ff; padding:10px; border:1px solid #007bff;">

  <input type="text" name="username" placeholder="Enter Name">

  <input type="submit" value="Submit GET" style="background-color:#007bff; color:white; padding:5px 10px; border:none; border-radius:4px;">

</form>


<br>


<!-- POST method form -->

<form method="post" style="background-color:#e9fbe9; padding:10px; border:1px solid #28a745;">

  <input type="text" name="email" placeholder="Enter Email">

  <input type="submit" value="Submit POST" style="background-color:#28a745; color:white; padding:5px 10px; border:none; border-radius:4px;">

</form>
```



b) Colors, border padding

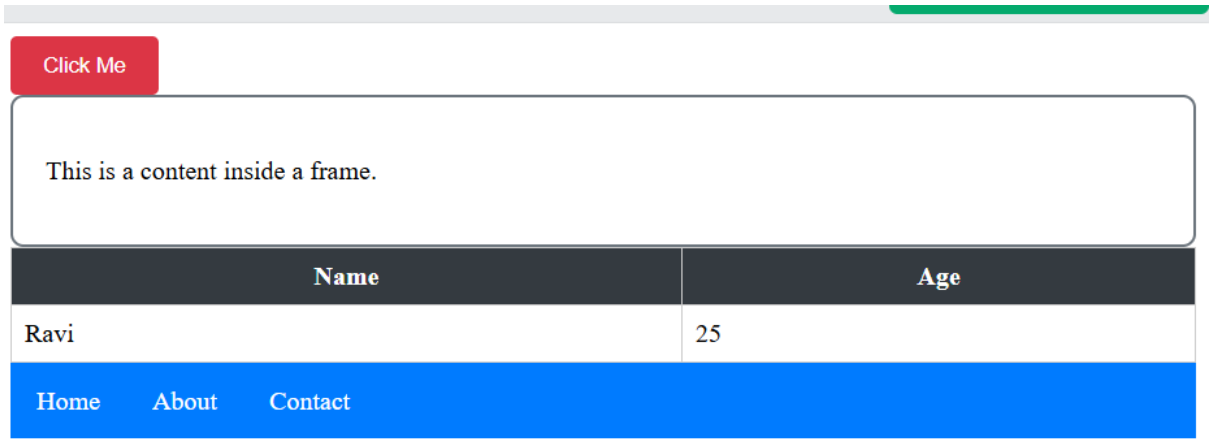
```
<div style="background-color:#fff3cd; border:1px solid #ffc107; padding:15px; border-radius:5px;">
  <p style="color:#856404; font-weight:bold;">This is a styled paragraph with border and padding.</p>
</div>
```



c) Buttons, frames, tables, nav bars

```
<button style="background-color:#dc3545; color:white; padding:10px 20px; border:none; border-radius:4px;">
  Click Me
</button>
<div style="border:2px solid #6c757d; padding:20px; border-radius:8px;">
  <p>This is a content inside a frame.</p>
</div>
<table style="border-collapse:collapse; width:100%;">
  <tr style="background-color:#343a40; color:white;">
    <th style="border:1px solid #ccc; padding:8px;">Name</th>
    <th style="border:1px solid #ccc; padding:8px;">Age</th>
  </tr>
  <tr>
    <td style="border:1px solid #ccc; padding:8px;">Ravi</td>
    <td style="border:1px solid #ccc; padding:8px;">25</td>
  </tr>
```

```
</table>

<div style="background-color:#007bff; overflow:hidden;">

  <a href="#" style="float:left; display:block; color:white; text-align:center; padding:14px 16px; text-decoration:none;">Home</a>

  <a href="#" style="float:left; display:block; color:white; text-align:center; padding:14px 16px; text-decoration:none;">About</a>

  <a href="#" style="float:left; display:block; color:white; text-align:center; padding:14px 16px; text-decoration:none;">Contact</a>

</div>
```



Experiment-8:
Implement web applications with Database using:
a) PHP
b) Servlets
c) JSP

Implementing web applications with database interaction using PHP, Servlets, and JSP involves distinct approaches for each technology.

a) PHP Web Application with Database:

- **Set up Environment:**

Install a web server (e.g., Apache), PHP, and a database (e.g., MySQL, PostgreSQL).

- **Database Connection:**

Establish a connection to the database using PHP's built-in database extensions (e.g., mysqli for MySQL, PDO for various databases).

```
<?php
$servername = "localhost";
$username = "root";
$password = "your_password";
$dbname = "your_database";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

- **Execute Queries:** Perform CRUD (Create, Read, Update, Delete) operations using SQL queries executed through the established connection.

```
<?php
// ... database connection code ...

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);
```

```
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

- **Display Data:** Present the retrieved data or results on HTML pages generated by PHP.

b) Servlets and JSP Web Application with Database:

- **Set up Environment:**

Install JDK, a Servlet/JSP container (e.g., Apache Tomcat), and a database.

- **Database Connection (JDBC):**

Use Java Database Connectivity (JDBC) to connect to the database within your Servlets.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseUtil {
    public static Connection getConnection() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/your_database";
        String user = "root";
        String password = "your_password";
        return DriverManager.getConnection(url, user, password);
    }
}
```

- **Servlet for Logic:** Servlets handle user requests, process data, interact with the database using JDBC, and prepare data to be displayed.

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

@WebServlet("/users")
public class UserServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        List<String> users = new ArrayList<>();
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT username FROM Users";
            try (PreparedStatement stmt = conn.prepareStatement(sql);
                 ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    users.add(rs.getString("username"));
                }
            }
        } catch (SQLException e) {
            throw new ServletException("Database error", e);
        }
        request.setAttribute("users", users);
```

```
        request.getRequestDispatcher("/WEB-INF/views/users.jsp").forward(request,
response);
    }
}
```

- **JSP for Presentation:** JSPs display the data retrieved by Servlets. Servlets forward requests to JSPs, passing data as request attributes.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
    <title>User List</title>
</head>
<body>
    <h1>Users</h1>
    <ul>
        <c:forEach var="user" items="${users}">
            <li>${user}</li>
        </c:forEach>
    </ul>
</body>
</html>
```

Experiment-9:
Design a simple calculator using:
a) JavaScript
b) PHP
c) Servlet
d) JSP

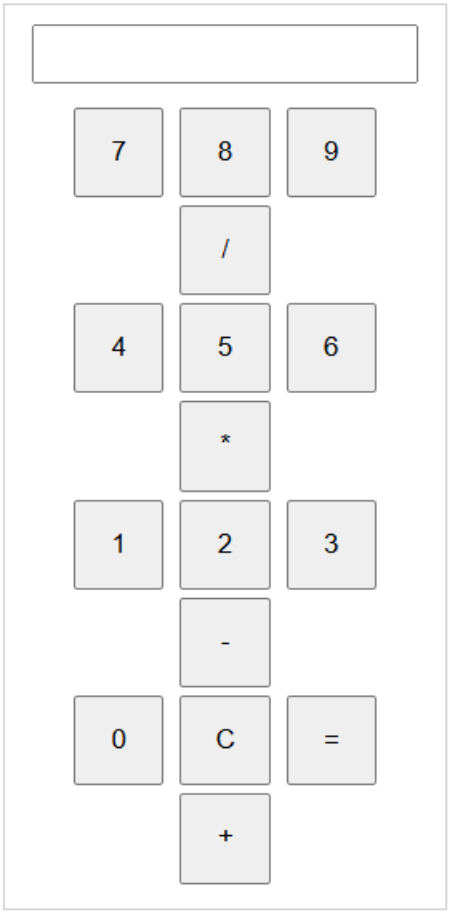
a) JavaScript

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple Calculator</title>
    <style>
        /* Basic CSS for styling the calculator layout */
        .calculator {
            width: 200px;
            margin: 50px auto;
            border: 1px solid #ccc;
            padding: 10px;
            text-align: center;
        }
        input[type="text"] {
            width: 90%;
            margin-bottom: 10px;
            padding: 5px;
        }
        button {
            width: 45px;
            height: 45px;
            margin: 2px;
        }
    </style>
</head>
```

```
<body>
    <div class="calculator">
        <input type="text" id="display" readonly>
        <div>
            <button onclick="appendNumber('7')">7</button>
            <button onclick="appendNumber('8')">8</button>
            <button onclick="appendNumber('9')">9</button>
            <button onclick="appendOperator('/')">/</button>
        </div>
        <div>
            <button onclick="appendNumber('4')">4</button>
            <button onclick="appendNumber('5')">5</button>
            <button onclick="appendNumber('6')">6</button>
            <button onclick="appendOperator('*')">*</button>
        </div>
        <div>
            <button onclick="appendNumber('1')">1</button>
            <button onclick="appendNumber('2')">2</button>
            <button onclick="appendNumber('3')">3</button>
            <button onclick="appendOperator('-')">-</button>
        </div>
        <div>
            <button onclick="appendNumber('0')">0</button>
            <button onclick="clearDisplay()">C</button>
            <button onclick="calculateResult()">=</button>
            <button onclick="appendOperator('+')">+</button>
        </div>
    </div>

    <script>
        let display = document.getElementById('display');
        let currentInput = '';
```

```
let operator = '';
let firstOperand = null;

function appendNumber(num) {
    currentInput += num;
    display.value = currentInput;
}

function appendOperator(op) {
    if (currentInput === '') return;
    if (firstOperand === null) {
        firstOperand = parseFloat(currentInput);
    } else if (operator) {
        calculateResult(); // Perform previous operation if one exists
    }
    operator = op;
    currentInput = '';
}

function calculateResult() {
    if (firstOperand === null || operator === '' || currentInput === '') return;
    let secondOperand = parseFloat(currentInput);
    let result;
    switch (operator) {
        case '+': result = firstOperand + secondOperand; break;
        case '-': result = firstOperand - secondOperand; break;
        case '*': result = firstOperand * secondOperand; break;
        case '/':
            if (secondOperand === 0) {
                alert("Cannot divide by zero!");
                clearDisplay();
                return;
```

```
                }
                result = firstOperand / secondOperand;
                break;
            default: return;
        }
        display.value = result;
        firstOperand = result;
        operator = '';
        currentInput = String(result); // Allow further operations on the result
    }

    function clearDisplay() {
        display.value = '';
        currentInput = '';
        operator = '';
        firstOperand = null;
    }
  </script>
</body>
</html>
```



b) PHP

A PHP calculator relies on form submissions and server-side processing.

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
    <title>PHP Calculator</title>
</head>
<body>
    <form action="calculate.php" method="post">
        <input type="text" name="num1" placeholder="Enter number 1" required>
        <select name="operator">
            <option value="+">+</option>
```

```
            <option value="-">-</option>
            <option value="*">*</option>
            <option value="/">/</option>
        </select>
        <input type="text" name="num2" placeholder="Enter number 2" required>
        <button type="submit">Calculate</button>
    </form>
</body>
</html>
<!-- calculate.php -->
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $num1 = $_POST['num1'];
    $num2 = $_POST['num2'];
    $operator = $_POST['operator'];
    $result = '';

    if (is_numeric($num1) && is_numeric($num2)) {
        switch ($operator) {
            case '+': $result = $num1 + $num2; break;
            case '-': $result = $num1 - $num2; break;
            case '*': $result = $num1 * $num2; break;
            case '/':
                if ($num2 != 0) {
                    $result = $num1 / $num2;
                } else {
                    $result = "Error: Division by zero!";
                }
                break;
            default: $result = "Invalid operator"; break;
        }
    } else {
```

```
        $result = "Invalid input: Please enter numbers.";
    }
    echo "Result: " . $result;
}
?>
```

c) Servlet

A Servlet-based calculator involves a web.xml configuration, an HTML form, and a Java Servlet class.

```
<!-- web.xml -->
<web-app>
    <servlet>
        <servlet-name>CalculatorServlet</servlet-name>
        <servlet-class>com.example.CalculatorServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>CalculatorServlet</servlet-name>
        <url-pattern>/calculate</url-pattern>
    </servlet-mapping>
</web-app>
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
    <title>Servlet Calculator</title>
</head>
<body>
    <form action="calculate" method="post">
        <input type="text" name="num1" placeholder="Enter number 1" required>
        <select name="operator">
            <option value="+">+</option>
            <option value="-">-</option>
            <option value="*">*</option>
```

```
            <option value="/">/</option>
        </select>
        <input type="text" name="num2" placeholder="Enter number 2" required>
        <button type="submit">Calculate</button>
    </form>
</body>
</html>
// com.example.CalculatorServlet.java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/calculate") // Alternative to web.xml mapping for modern servlets
public class CalculatorServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        try {
            double num1 = Double.parseDouble(request.getParameter("num1"));
            double num2 = Double.parseDouble(request.getParameter("num2"));
            String operator = request.getParameter("operator");
            double result = 0;

            switch (operator) {
                case "+": result = num1 + num2; break;
                case "-": result = num1 - num2; break;
                case "*": result = num1 * num2; break;
```

```
            case "/":
                if (num2 == 0) {
                    out.println("Error: Division by zero!");
                    return;
                }
                result = num1 / num2;
                break;
            default: out.println("Invalid operator"); return;
        }
        out.println("Result: " + result);

    } catch (NumberFormatException e) {
        out.println("Invalid input: Please enter numbers.");
    } finally {
        out.close();
    }
  }
}
```

d) JSP

A simple calculator using JSP typically involves two main components: an HTML form for user input and a JSP page to process the calculation.

1. index.jsp (Input Form):

This JSP file will present the user with input fields for two numbers and buttons for the desired operations.

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple JSP Calculator</title>
</head>
<body>
    <h2>Simple Calculator</h2>
    <form action="calculate.jsp" method="post">
```

```
        Number 1: <input type="text" name="num1"><br><br>
        Number 2: <input type="text" name="num2"><br><br>
        <input type="submit" name="operation" value="Add">
        <input type="submit" name="operation" value="Subtract">
        <input type="submit" name="operation" value="Multiply">
        <input type="submit" name="operation" value="Divide">
    </form>
</body>
</html>
```

2. calculate.jsp (Calculation Logic):

This JSP file will receive the input from index.jsp, perform the calculation based on the chosen operation, and display the result.

Experiment-10:
Create registration and login forms with validations using JScript query.

```
<!DOCTYPE html>
<html>
<head>
<title>Registration Form Using jQuery - Demo Preview</title>
<meta name="robots" content="noindex, nofollow">
<!-- Include CSS File Here -->
<link rel="stylesheet" href="css/style.css"/>
<!-- Include JS File Here -->
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
<script type="text/javascript" src="js/registration.js"></script>
</head>
<body>
<div class="container">
<div class="main">
<form class="form" method="post" action="#">
<h2>Create Registration Form Using jQuery</h2>
<label>Name :</label>
<input type="text" name="dname" id="name">
<label>Email :</label>
<input type="text" name="demail" id="email">
<label>Password :</label>
<input type="password" name="password" id="password">
<label>Confirm Password :</label>
<input type="password" name="cpassword" id="cpassword">
<input type="button" name="register" id="register" value="Register">
</form>
</div>
</body>
</html>
```

jQuery File: registration.js

Given below our complete jQuery code.

```
$(document).ready(function() {
$("#register").click(function() {
var name = $("#name").val();
var email = $("#email").val();
var password = $("#password").val();
var cpassword = $("#cpassword").val();
if (name == '' || email == '' || password == '' || cpassword == '') {
alert("Please fill all fields...!!!!!!");
} else if ((password.length) < 8) {
alert("Password should atleast 8 character in length...!!!!!!");
} else if (!(password).match(cpassword)) {
alert("Your passwords don't match. Try again?");
} else {
$.post("register.php", {
name1: name,
email1: email,
password1: password
}, function(data) {
if (data == 'You have Successfully Registered.....') {
$("form")[0].reset();
}
alert(data);
});
}
});
});
```

Copy

PHP Script: register.php

Below script executes for inserting data into database.

```
<?php
$connection = mysql_connect("localhost", "root", ""); // Establishing connection with server..
```

```
$db = mysql_select_db("college", $connection); // Selecting Database.

$name=$_POST['name1']; // Fetching Values from URL.

$email=$_POST['email1'];

$password= sha1($_POST['password1']); // Password Encryption, If you like you can also leave sha1.

// Check if e-mail address syntax is valid or not

$email = filter_var($email, FILTER_SANITIZE_EMAIL); // Sanitizing email(Remove unexpected symbol like <,>,?,#,!, etc.)

if (!filter_var($email, FILTER_VALIDATE_EMAIL)){

echo "Invalid Email.......";

}else{

$result = mysql_query("SELECT * FROM registration WHERE email='$email'");

$data = mysql_num_rows($result);

if(($data)==0){

$query = mysql_query("insert into registration(name, email, password) values ('$name', '$email', '$password')"); // Insert query

if($query){

echo "You have Successfully Registered.....";

}else

{

echo "Error....!!";

}

}else{

echo "This email is already registered, Please try another email...";

}

}

mysql_close ($connection);

?>
```

Copy

My SQL Code Segment:

To create database and table, execute following codes in your My SQL .

```
CREATE DATABASE college;
```

```
CREATE TABLE registration(
id int(10) NOT NULL AUTO_INCREMENT,
name varchar(255) NOT NULL,
email varchar(255) NOT NULL,
password varchar(255) NOT NULL,
PRIMARY KEY (id)
)
```

Copy

CSS File: style.css

Styling HTML Elements.

```
/* Below line is used for online Google font */
@import url(http://fonts.googleapis.com/css?family=Droid+Serif);
h2{
text-align: center;
font-size: 24px;
}
hr{
margin-bottom: 30px;
}
div.container{
width: 960px;
height: 610px;
margin:50px auto;
font-family: 'Droid Serif', serif;
position:relative;
}
div.main{
width: 320px;
float:left;
padding: 10px 55px 40px;
background-color: rgba(187, 255, 184, 0.65);
border: 15px solid white;
```

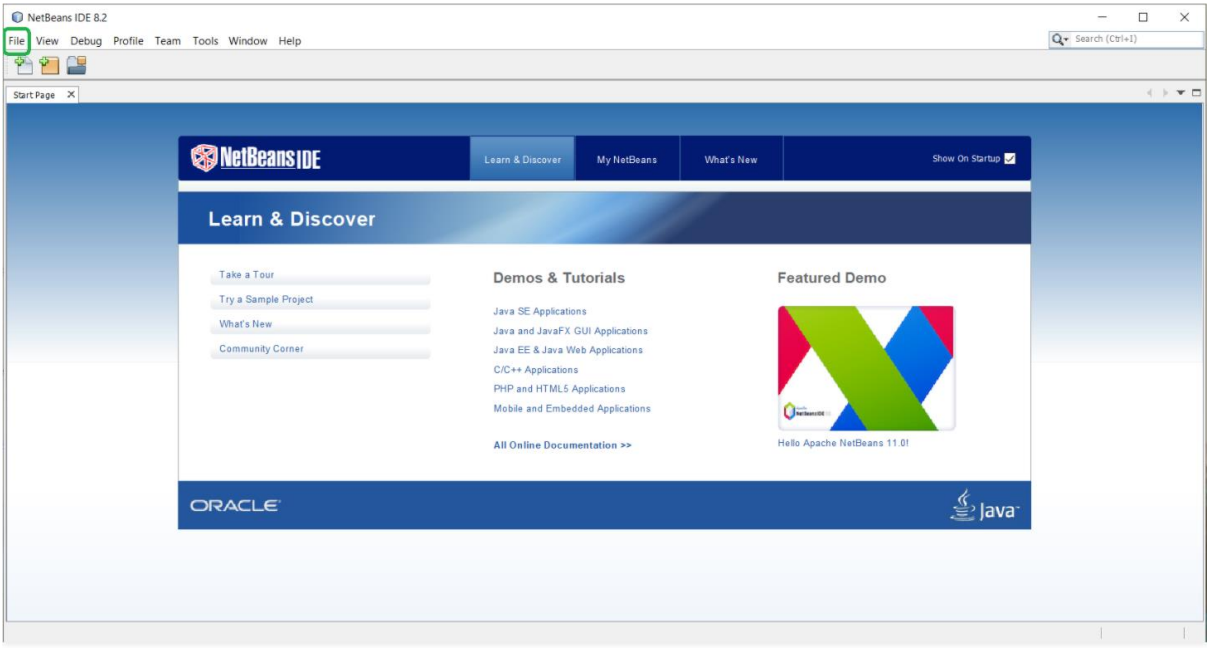
```
box-shadow: 0 0 10px;
border-radius: 2px;
font-size: 13px;
}
input[type=text],[type=password] {
width: 97.7%;
height: 34px;
padding-left: 5px;
margin-bottom: 20px;
margin-top: 8px;
box-shadow: 0 0 5px #00F5FF;
border: 2px solid #00F5FF;
color: #4f4f4f;
font-size: 16px;
}
label{
color: #464646;
text-shadow: 0 1px 0 #fff;
font-size: 14px;
font-weight: bold;
}
#register {
font-size: 20px;
margin-top: 15px;
background: linear-gradient(#22abe9 5%, #36caf0 100%);
border: 1px solid #0F799E;
padding: 7px 35px;
color: white;
text-shadow: 0px 1px 0px #13506D;
font-weight: bold;
border-radius: 2px;
cursor: pointer;
```

```
width: 100%;
}
#register:hover{
background: linear-gradient(#36caf0 5%, #22abe9 100%);
}
```

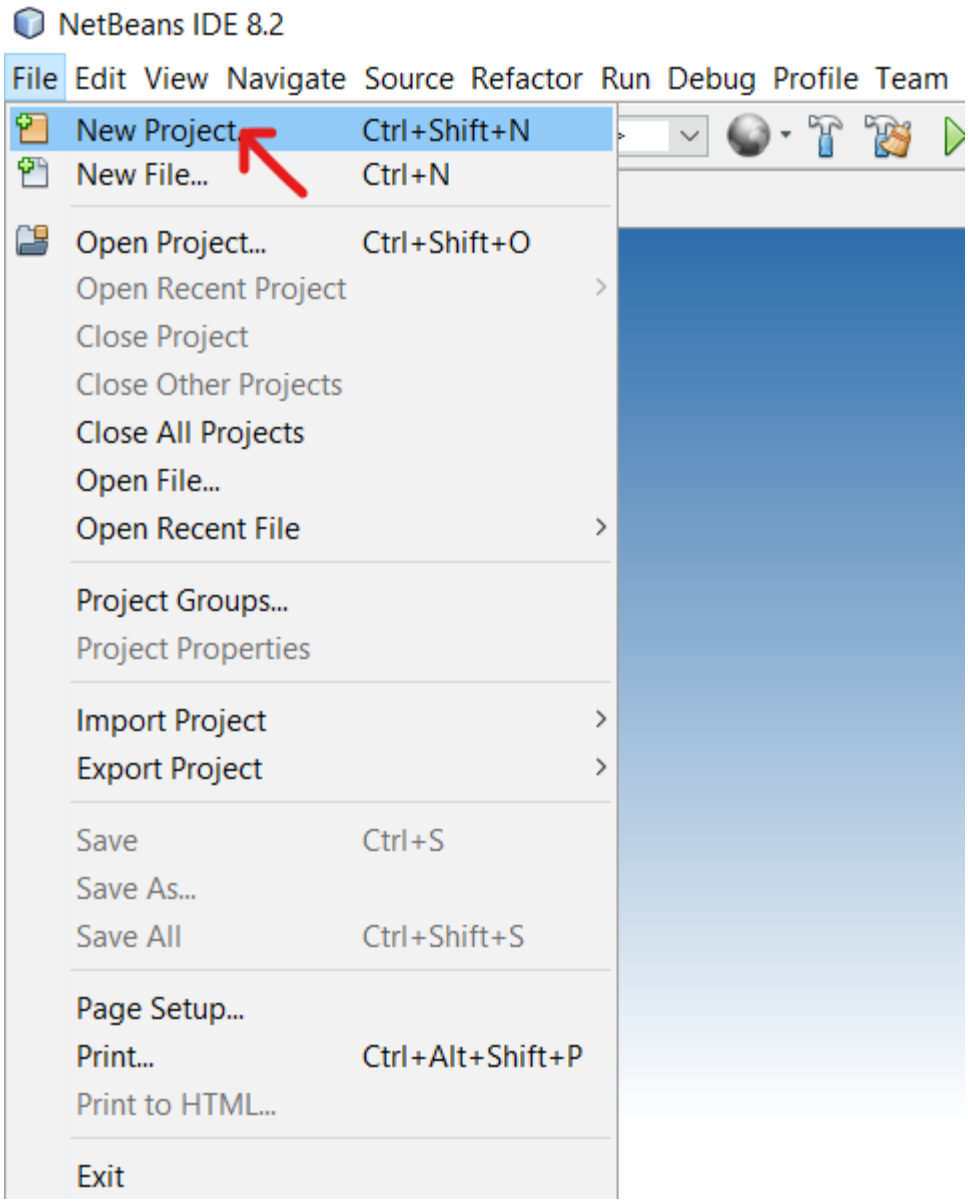
Experiment-11:
JScript to retrieve student information from student database using database connectivity.

Steps to create the application:

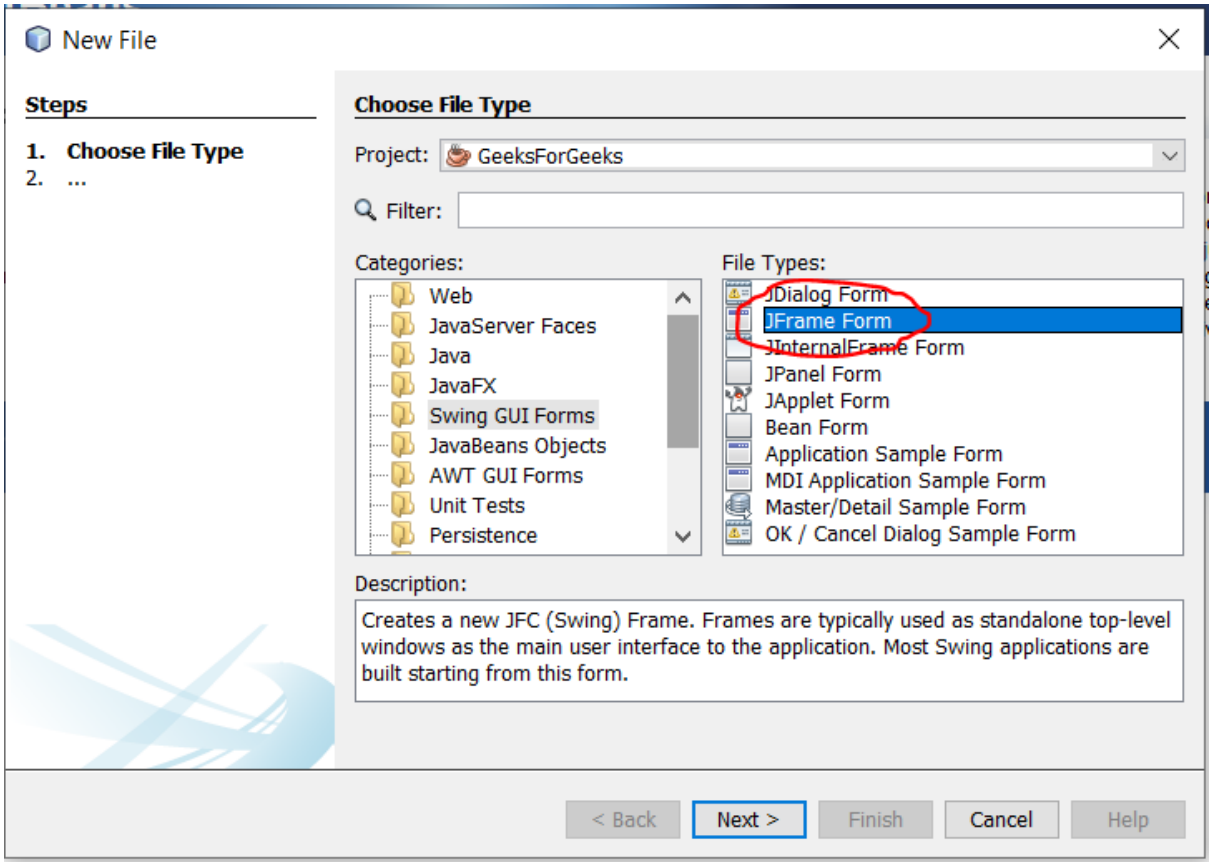
1. First, open Netbeans and click on the File option from the menu bar.



2. Now create a new Java application by clicking on **New Project -> Java -> Java Application** and give a suitable project name and click finish.

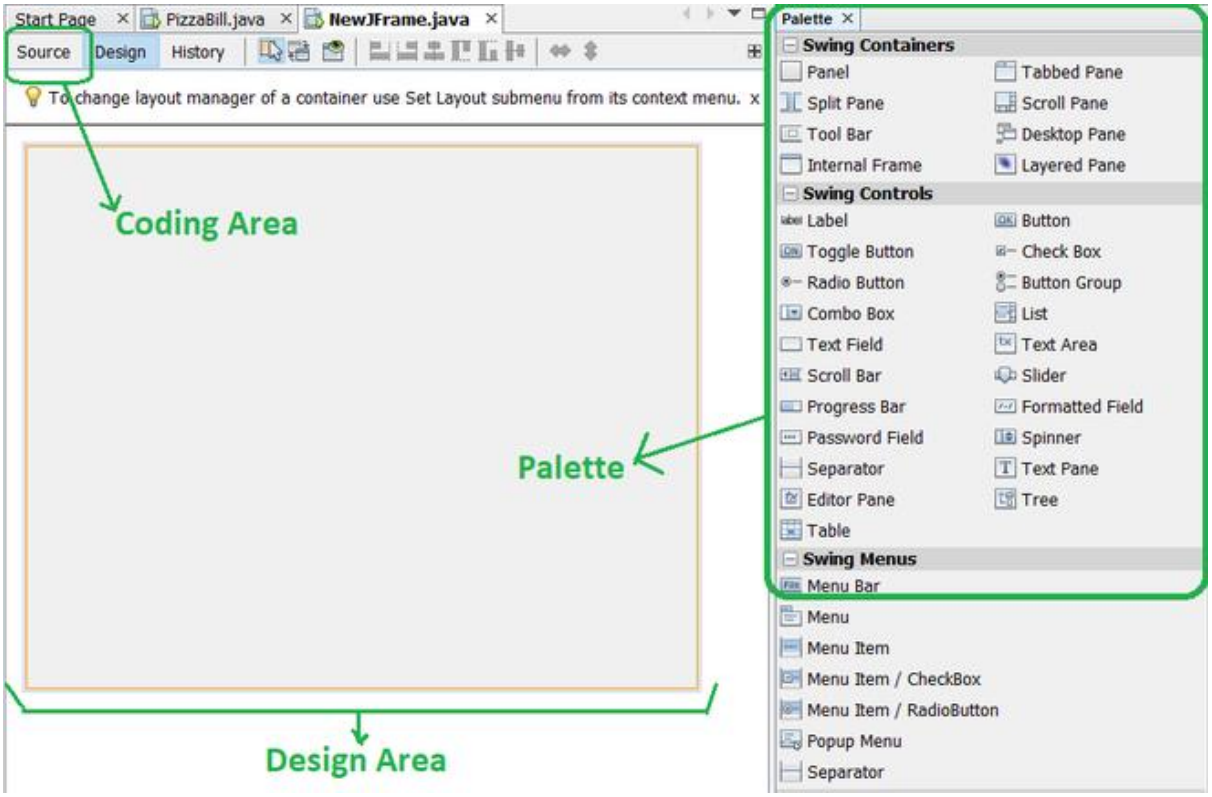


3. Now create a new file by going to the File option again on the menu bar, then **New File -> Swing GUI Forms -> JFrame Form**, and give a suitable file name click finish.



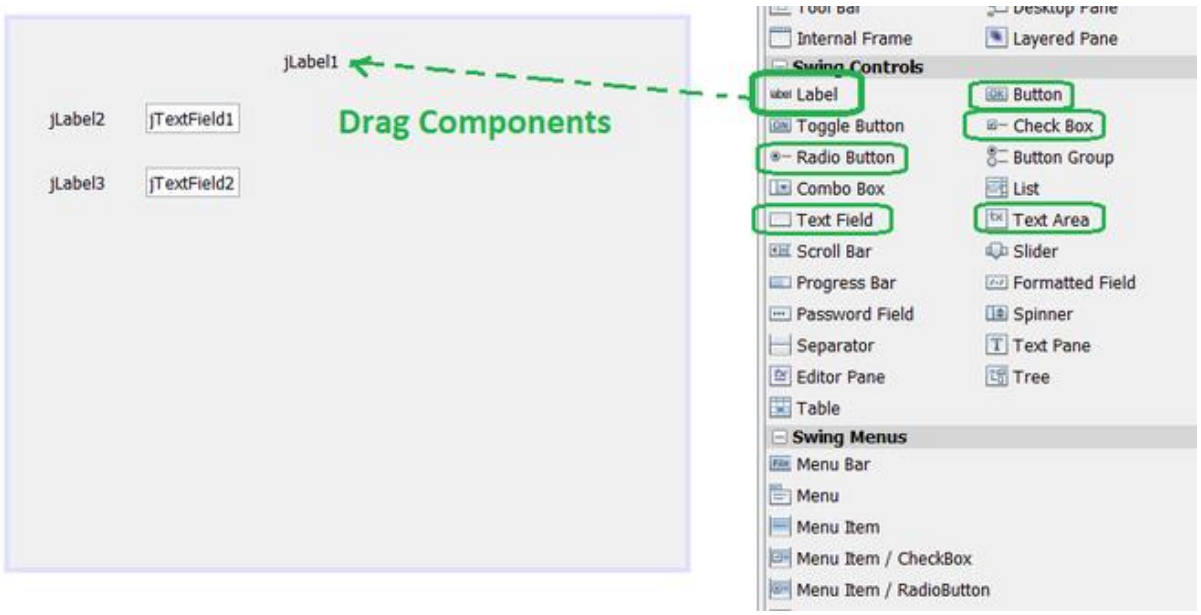
4. After successful file creation, we will now be presented with the following screen. The 3 important parts of this window are:

- **Design:** This is the area where we will create the design/template of our application.
- **Source:** This is where the logic code of the program is written.
- **Palette:** This component contains all the widgets which we need to drag and drop on the design area



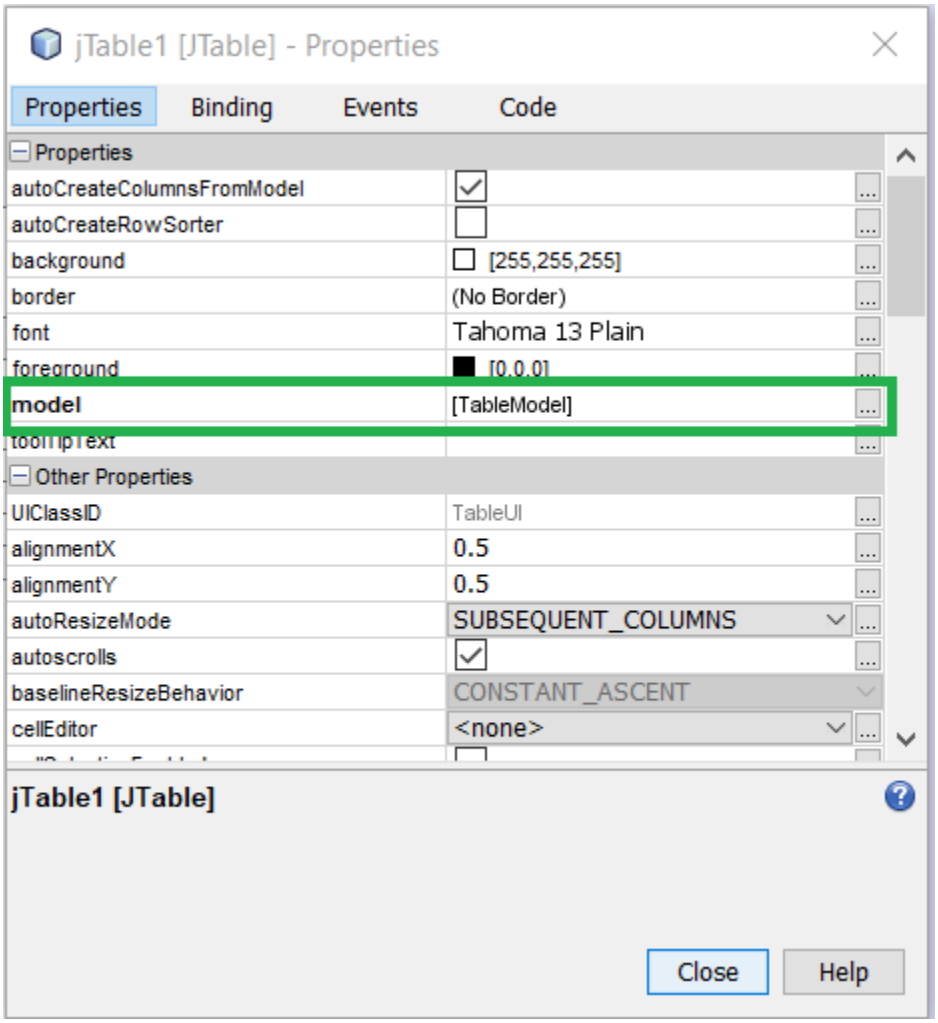
Window displayed after successful Java file creation.

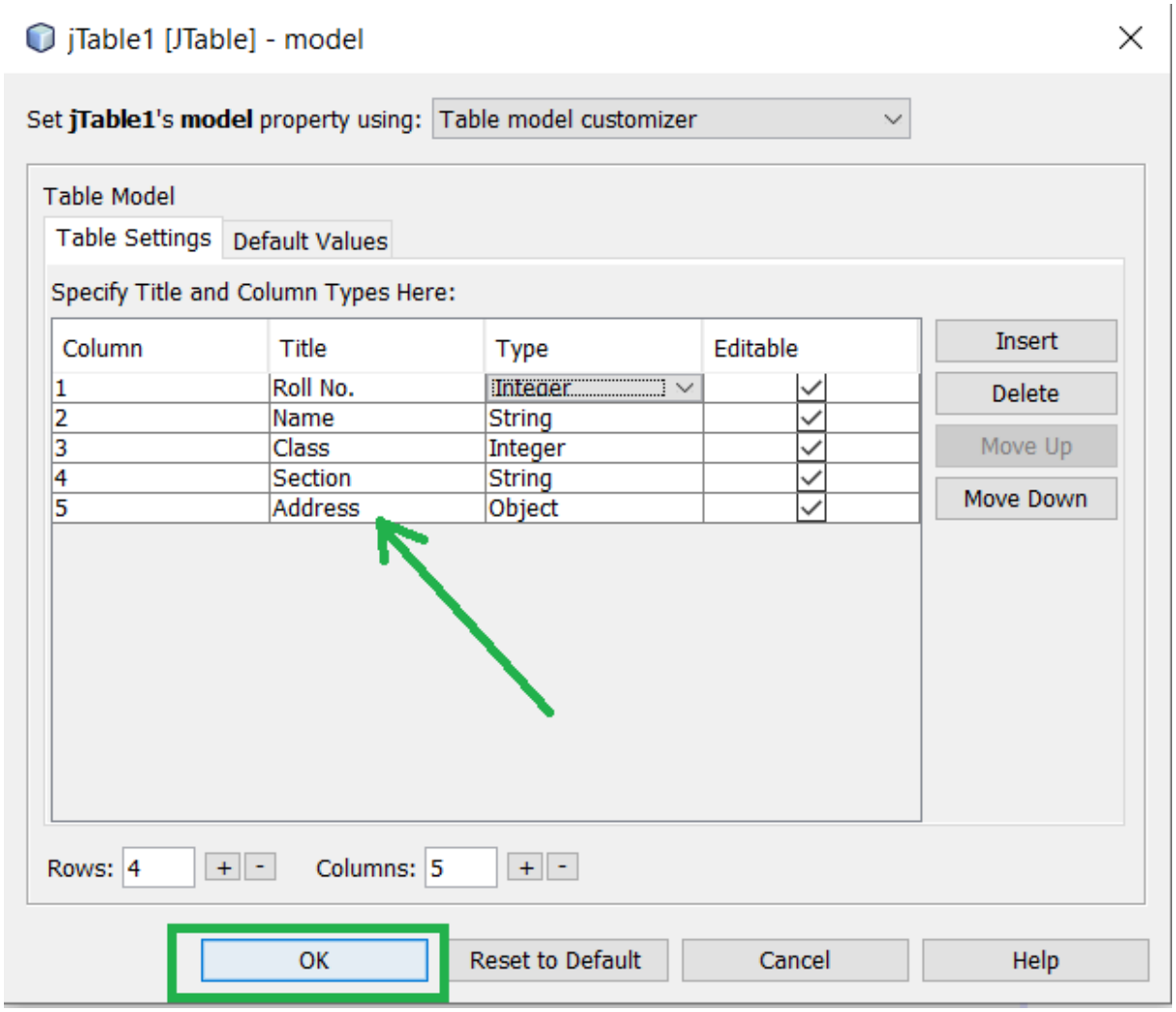
5. Now from the palette situated at the right-hand side of the window, start dragging the toolkit widgets.



Drag Components from palette to design area

6. Since we need to display all data in a tabulated form, drag the table widget from the palette onto the design area. Now to set the headings, right-click on the table, select **Properties -> Model ->Add/delete Columns**.


jTable1 [JTable] - Properties
Properties
Binding
Events
Code
Properties
autoCreateColumnsFromModel
autoCreateRowSorter
background
[255,255,255]
border
(No Border)
font
Tahoma 13 Plain
foreground
[0,0,0]
model
[TableModel]
toolTipText
Other Properties
UIClassID
TableUI
alignmentX
0.5
alignmentY
0.5
autoResizeMode
SUBSEQUENT_COLUMNS
autoscrolls
baselineResizeBehavior
CONSTANT_ASCENT
cellEditor
<none>
jTable1 [JTable]
Close
Help



7. Now let us create the database to store the data. Open MySQL command client, enter password, and type in the following commands to create a new database, new table, and defining the attributes.

mysql> create database student;

Query OK, 1 row affected (0.14 sec)

mysql> use student;

Database changed

mysql> create table record(

-> rollno int(3),

-> name char(20),

-> class int(2),

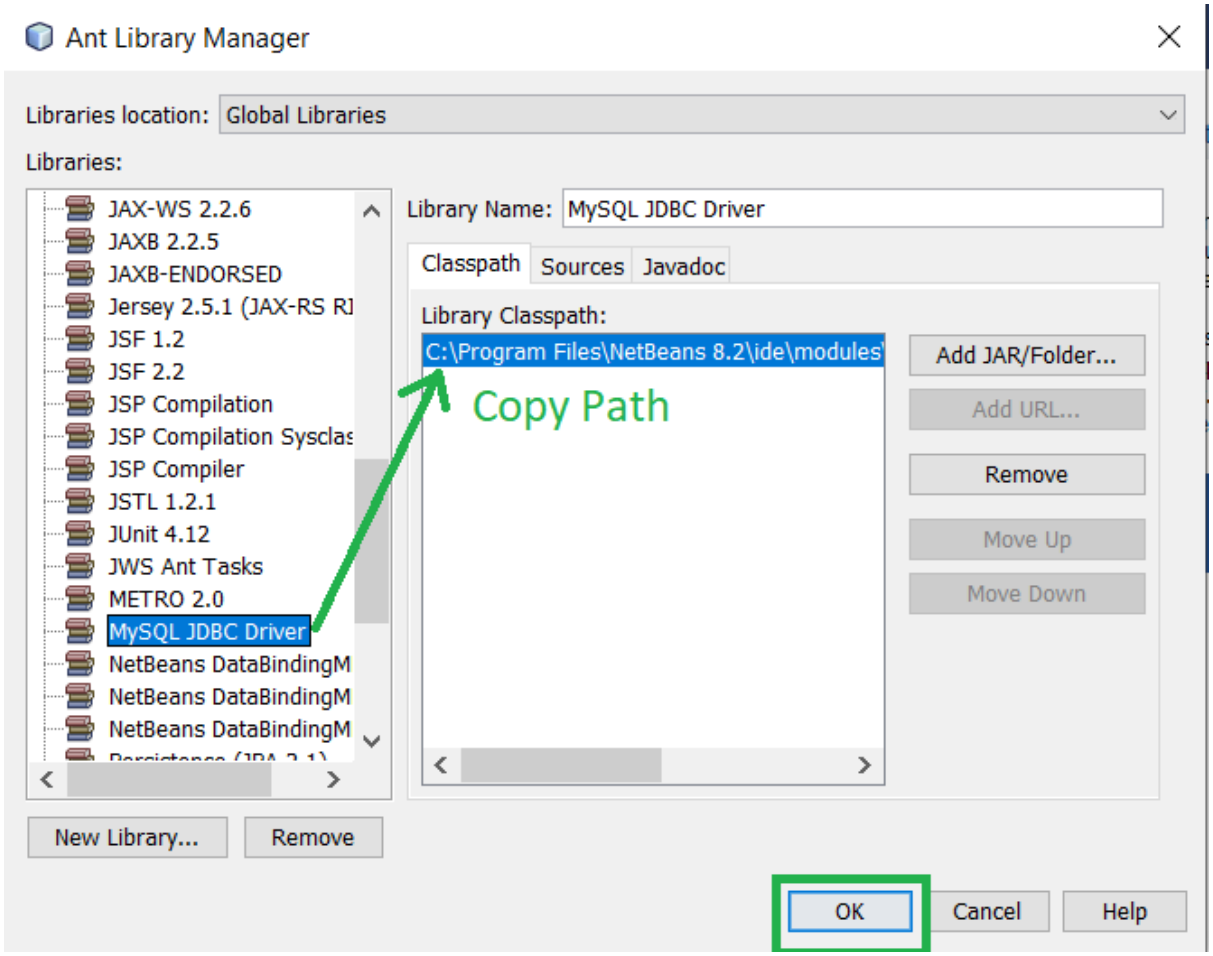
-> section char(2),

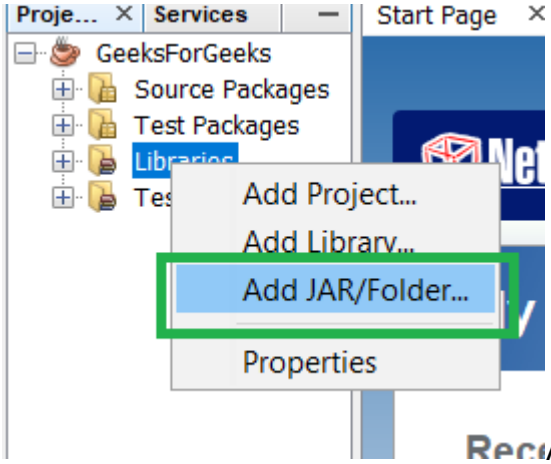
-> address varchar(40));

Query OK, 0 rows affected (2.03 sec)

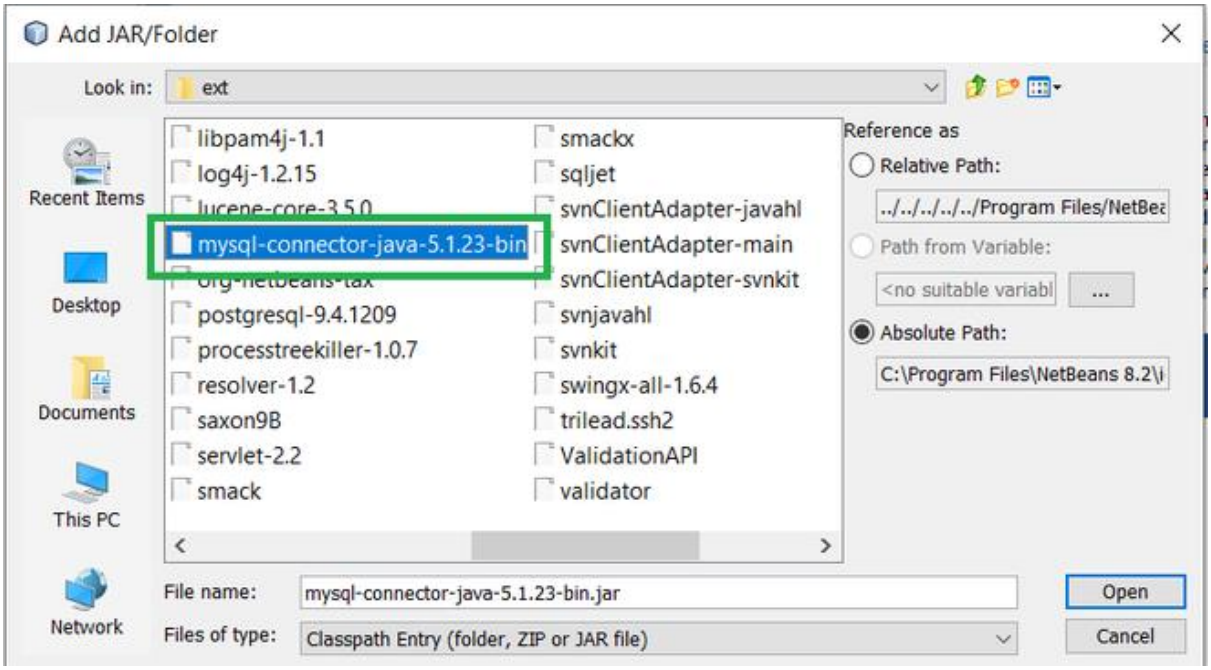
```
mysql> describe record;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| rollno  | int(3)      | YES  |     | NULL    |       |
| name    | char(20)    | YES  |     | NULL    |       |
| class   | int(2)      | YES  |     | NULL    |       |
| section | char(2)     | YES  |     | NULL    |       |
| address | varchar(40) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.03 sec)
```

8. We need to import libraries that are needed to set up a connection with the database and retrieve data which is done by - *DriverManager Class*, *Connection Class*, and *Statement Class*. Thus go to the menubar, under **Tools->Libraries**, and add *MySQL JDBC connector*. Note down the Library ClassPath and click OK. Now go to **Projects** toolbar and go to your application's Libraries. Right-click and select **Add Jar/Library** and browse the Library classpath noted down previously.


Ant Library Manager
Libraries location:
Global Libraries
Libraries:
JAX-WS 2.2.6
JAXB 2.2.5
JAXB-ENDORSED
Jersey 2.5.1 (JAX-RS RI
JSF 1.2
JSF 2.2
JSP Compilation
JSP Compilation Sysclas
JSP Compiler
JSTL 1.2.1
JUnit 4.12
JWS Ant Tasks
METRO 2.0
MySQL JDBC Driver
NetBeans DataBindingM
NetBeans DataBindingM
NetBeans DataBindingM
Library Name:
MySQL JDBC Driver
Classpath
Sources
Javadoc
Library Classpath:
C:\Program Files\NetBeans 8.2\ide\modules
Copy Path
Add JAR/Folder...
Add URL...
Remove
Move Up
Move Down
New Library...
Remove
OK
Cancel
Help

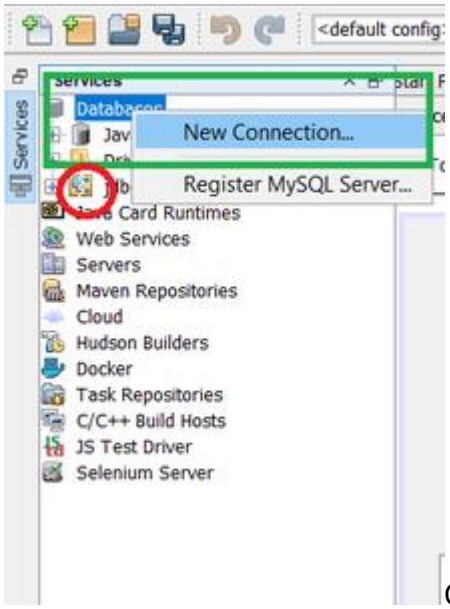


Add Jar file

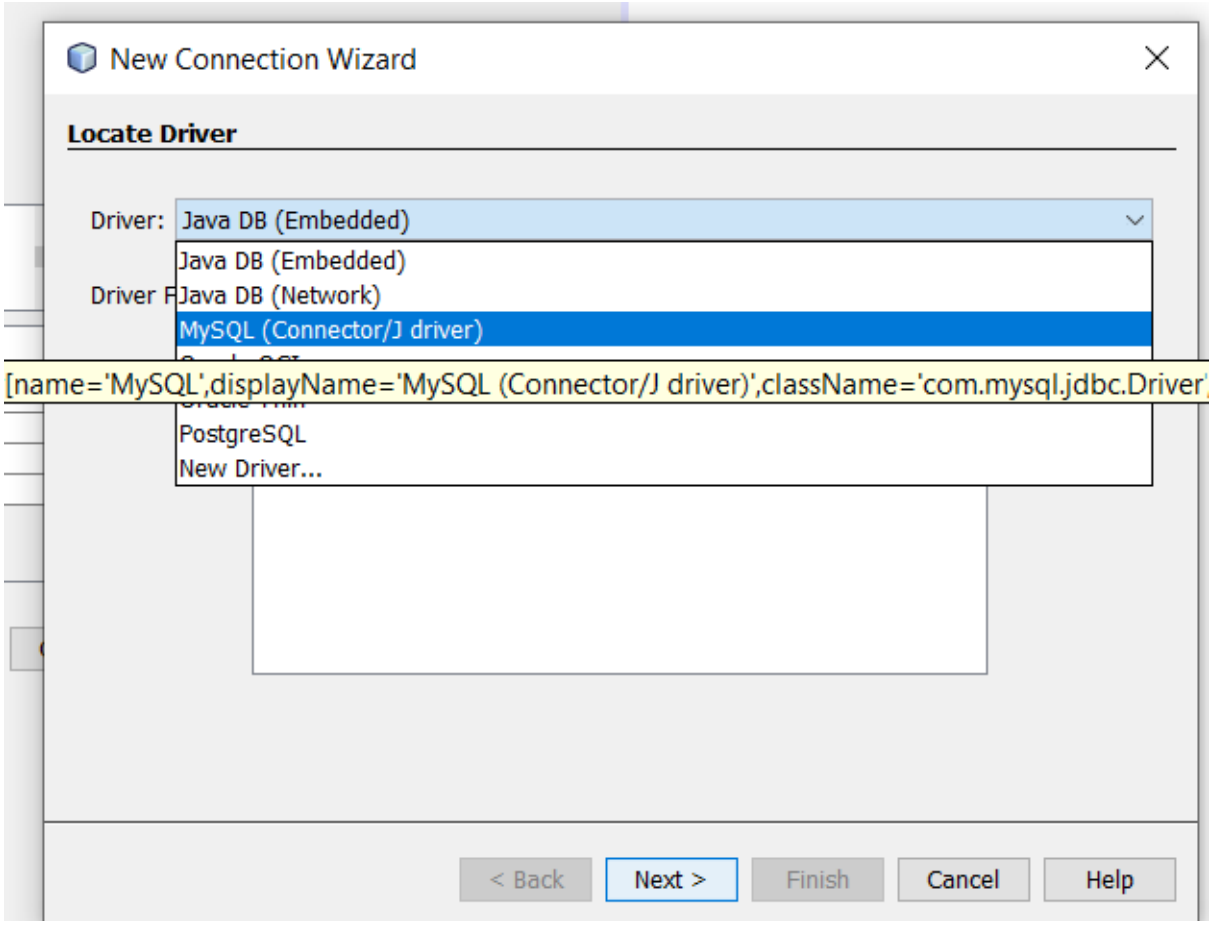


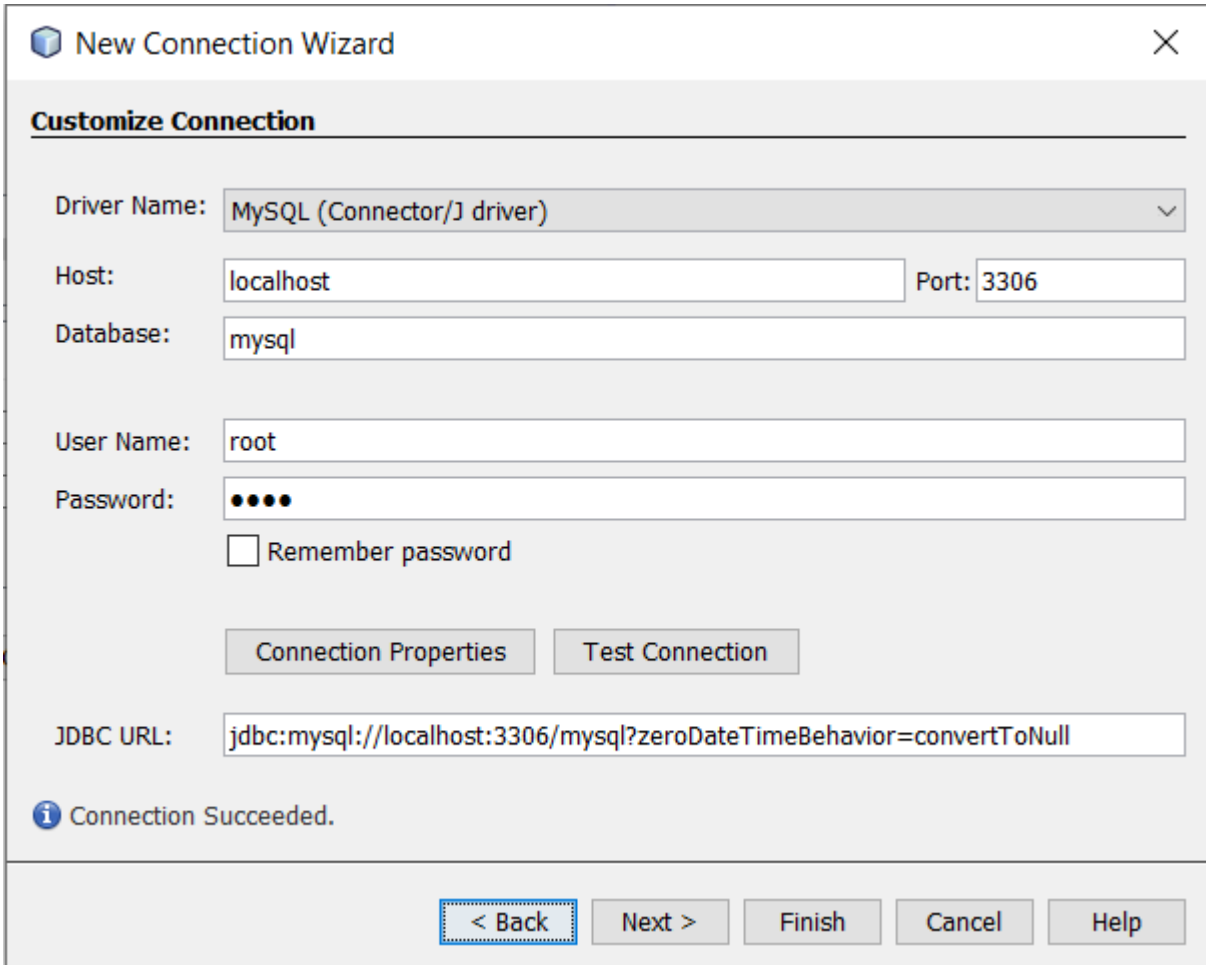
Add mysql connector

9. Go to **Windows->Services->Databases** and enter the required credentials of your MySQL username and password. After clicking test connection, if it's successful, the connector logo appears connected.



Connection is initially broken


New Connection Wizard
Locate Driver
Driver:
Java DB (Embedded)
Java DB (Embedded)
Java DB (Network)
MySQL (Connector/J driver)
[name='MySQL',displayName='MySQL (Connector/J driver)',className='com.mysql.jdbc.Driver'
PostgreSQL
New Driver...
< Back
Next >
Finish
Cancel
Help



10. Now to type the code, double-click on jButton1 (Insert), you will be directed to the source tab. Here type in the following code.

```
// Write the import code at top
import java.sql.*;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;


public class SchoolRecord extends javax.swing.JFrame {


    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
    // JButton1 Code starts from here
    try {
        Class.forName("com.mysql.jdbc.Driver");
```

```
        Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/student", "root",
            "root");
        stmt = con.createStatement();
        String rollno = jTextField1.getText();
        String name = jTextField2.getText();
        String class = jTextField3.getText();
        String sec = jTextField4.getText();
        String adr = jTextArea1.getText();
        String INSERT = "INSERT INTO RECORD VALUES('"
                    + rollno + "','" + name + "','"
                    + class + "','" + sec + "','" + adr
                    + "');";
        stmt.executeUpdate(INSERT);
        JOptionPane.showMessageDialog(
            this, "Record Added Successfully");
        jButton1.setEnabled(true);
    }
    catch (Exception e) {
        JOptionPane.showMessageDialog(
            this, "Error In Connectivity");
    }
```

In the above code, the following things need to be kept in mind which are:

1. **Connection Class:** It acts as a connection session between the Java program and specific database application. It is through which we send SQL queries to the database.
2. **Statement Class:** A Statement is an interface that represents a SQL statement.
3. **ResultSet:** When you execute Statement objects, and they generate ResultSet objects, which is a table of data representing a database result set. A Connection object is needed to create a Statement object.
4. **JDBC Driver Registration:** To open a connection to the database from Java application, the JDBC driver should be registered with the Device Manager. Hence we use **forName()** of language package. Here *com.mysql.jdbc.Driver* is the driver name for MySQL.

5. **.getConnection():** It is used to establish a physical connection to the database by specifying the *database name(student) , username(root) and password(root).* This creates a connection object.
6. **Query Execution:** *createStatement()* creates a statement type object that holds SQL queries. Then *executeQuery/executeUpdate* method executes the SQL statement. Here it is "INSERT INTO RECORD VALUES....".
7. **Data Extraction:** The above method creates a resultset object that contains the resultant data. *(Now see the code below)* rs is the variable that stores the resultant dataset and hence we use a *.get<Type>()* method to obtain data.
8. **while(rs.next()):** *(See code below)* Since we need data containing multiple rows, we use a loop to access them. The *next()* method moves the cursor forward by one row.
9. **Closing open databases:** We close all open databases to clean up the environment. Thus we use- *rs.close() , stmt.close() and con.close()* methods.

11. Now to display all data, write the following code under the jButton2 ActionPerformed option which can be achieved by clicking twice on *View Data Button* in the design area.

```
DefaultTableModel model
   = (DefaultTableModel)jTable1.getModel();
try {
   Class.forName("com.mysql.jdbc.Driver");
   Connection con = DriverManager.getConnection(
      "jdbc:mysql://localhost:3306/student", "root",
      "root");
   String query = "SELECT* FROM RECORD;";
   ResultSet rs = stmt.executeQuery(query);
   while (rs.next()) {
      String rollno = rs.getString("rollno");
      String name = rs.getString("name");
      String class = rs.getString("class");
      String sec = rs.getString("section");
      String adr = rs.getString("address");
      model.addRow(
         new Object[] { rollno, name, class, sec, adr });
   }
   rs.close();
   stmt.close();
```

```
    con.close();
}
catch (Exception e) {
    JOptionPane.showMessageDialog(this,
                        "Error In Connectivity");
}
```

12. Now to clear all textfields, textareas and table content write the following code under the jButton3 ActionPerformed option which can be achieved by clicking twice on Clear Button in the design area.

```
jTextField1.setText("");
jTextField2.setText("");
jTextField3.setText("");
jTextField4.setText("");
jTextArea1.setText("");
DefaultTableModel dm
    = (DefaultTableModel)jTable1.getModel();
dm.getDataVector().removeAllElements();
jTable1.repaint();
```

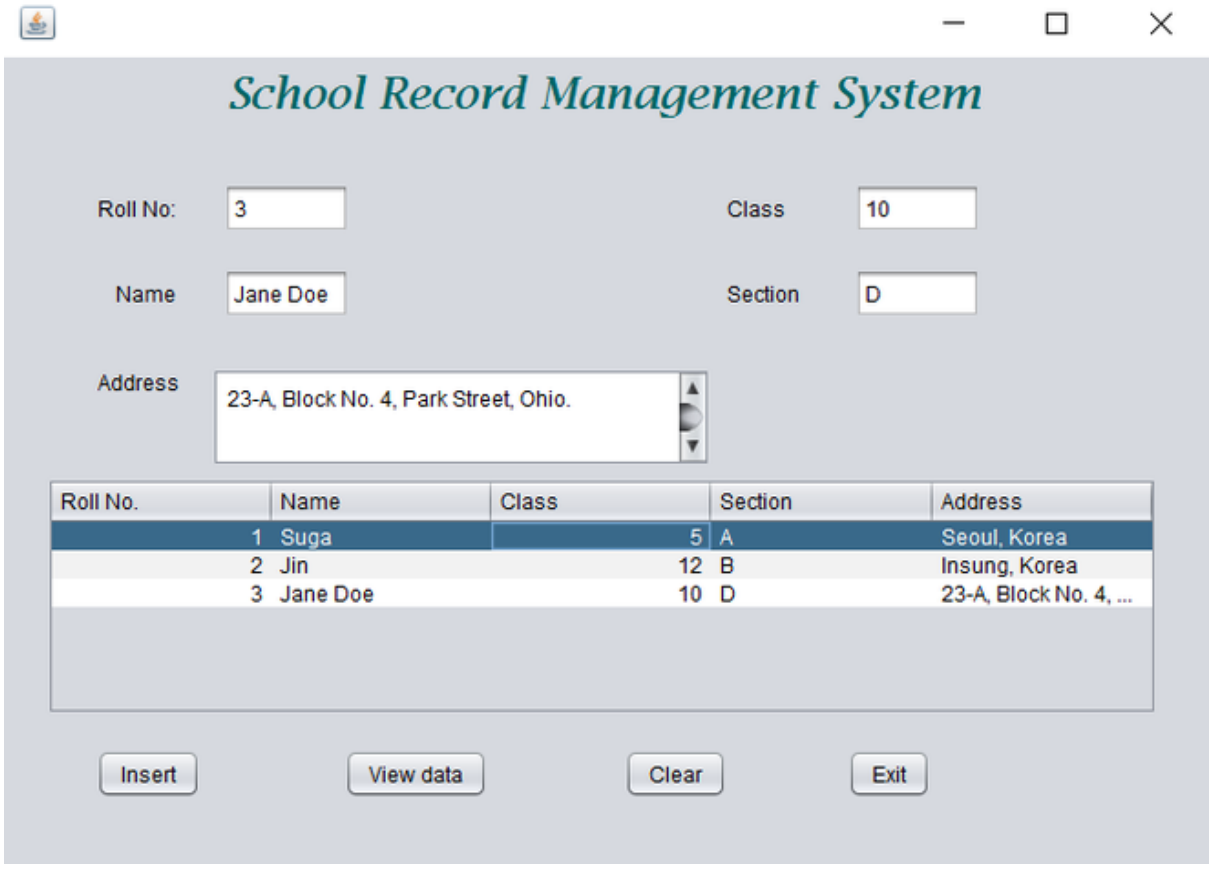
13. Now to exit from the system, add the following statement under the jButton4 ActionPerformed option which can be achieved by clicking twice on Exit Button in the design area.

```
System.exit(0);
```

14. After code is typed, right-click anywhere on the screen and select the Run File option from the drop-down menu. The final output is shown below. Input necessary details and the application is ready!

School Record Management System
Roll No:
1
Class
5
Name
Suga
Section
A
Address
Seoul, Korea
Roll No.
Name
Address
Message
Record Added Successfully
OK
Insert
View data
Clear
Exit

Insert Data


School Record Management System
Roll No:
3
Class
10
Name
Jane Doe
Section
D
Address
23-A, Block No. 4, Park Street, Ohio.
Roll No.
Name
Class
Section
Address
1 Suga 5 A Seoul, Korea
2 Jin 12 B Insung, Korea
3 Jane Doe 10 D 23-A, Block No. 4, ...
Insert
View data
Clear
Exit

Experiment-12:
Implement in React JS:
a) Create constructs and data elements
b) Implement DOM operations

```
import React, { useState } from 'react';

function App() {
  // Data Element: List of names (array)
  const [names, setNames] = useState(["Sai", "Kiran"]);

  // Data Element: Input value (string)
  const [inputValue, setInputValue] = useState("");

  // DOM Operation: Add name to list
  const handleAdd = () => {
    if (inputValue.trim() !== "") {
      setNames([...names, inputValue]);
      setInputValue(""); // Clear input box
    }
  };

  // DOM Operation: Remove a name
  const handleRemove = (index) => {
    const updatedNames = [...names];
    updatedNames.splice(index, 1);
    setNames(updatedNames);
  };

  return (
    <div style={{ padding: '20px' }}>
      <h2>Simple React Example</h2>
```

```
      {/* Input Field to enter name */}
      <input
        type="text"
        value={inputValue}
        onChange={(e) => setInputValue(e.target.value)}
        placeholder="Enter name"
      />

      {/* Button to add name */}
      <button onClick={handleAdd}>Add</button>

      {/* Displaying list of names */}
      <ul>
        {names.map((name, index) => (
          <li key={index}>
            {name}
            <button style={{ marginLeft: '10px' }} onClick={() => handleRemove(index)}>
              Remove
            </button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

Simple React Example

Enter name Add

- Sai Remove
- Kiran Remove

Experiment-13:
Implement in Angular JS:
a) Data binding
b) Directives and events
c) Fetching data from MySQL

◈ Step 1: HTML + AngularJS Frontend

Save this as index.html
(AngularJS CDN is used here, not Angular 2+)

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <title>AngularJS MySQL Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.3/angular.min.js"></script>
</head>
<body ng-controller="myCtrl">

  <h2>Student Data</h2>

  <!-- a) Data Binding -->
  <label>Enter Name:</label>
  <input type="text" ng-model="studentName">
  <p>You typed: {{ studentName }}</p>

  <!-- b) Directive & Event -->
  <button ng-click="loadData()">Load Data from MySQL</button>

  <table border="1" style="margin-top: 10px;">
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Course</th>
    </tr>
```

```
    <tr ng-repeat="student in students">
      <td>{{ student.id }}</td>
      <td>{{ student.name }}</td>
      <td>{{ student.course }}</td>
    </tr>
  </table>

  <script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function($scope, $http) {
      $scope.studentName = "";

      // c) Fetching data from MySQL
      $scope.loadData = function() {
        $http.get("fetch.php")
        .then(function(response) {
          $scope.students = response.data;
        });
      };
    });
  </script>

</body>
</html>
```

◈ Step 2: PHP Backend to Fetch from MySQL

Save this as fetch.php (in the same folder on your local server like XAMPP)

```
<?php
$host = "localhost";
$user = "root";
$password = "";
$db = "college"; // make sure this DB exists
```

```
$conn = new mysqli($host, $user, $password, $db);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, name, course FROM students";
$result = $conn->query($sql);

$data = array();

while($row = $result->fetch_assoc()) {
    $data[] = $row;
}

echo json_encode($data);
$conn->close();
?>
```

🔷 Step 3: Sample MySQL Table

Make sure you create a MySQL table like this:

```
CREATE DATABASE college;

USE college;

CREATE TABLE students (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  course VARCHAR(50)
);

INSERT INTO students (name, course) VALUES
```

('Sai', 'M.Tech'),

('Kiran', 'MBA'),

('Anjali', 'B.Tech');

Experiment-14:
Invoke data using JScript from MongoDB.

To invoke and retrieve data from MongoDB using JavaScript, you typically use the official MongoDB Node.js driver. This involves setting up a Node.js application and connecting to your MongoDB database.

Here's a general outline of the process: Install the MongoDB Node.js Driver.

```
npm install mongodb
```

Connect to MongoDB.

```
const { MongoClient } = require('mongodb');


// Replace with your MongoDB connection string
const uri = "mongodb+srv://<username>:<password>@<cluster-url>/<database-name>?retryWrites=true&w=majority";


const client = new MongoClient(uri);


async function connectAndOperate() {
    try {
        await client.connect();
        console.log("Connected to MongoDB!");


        const database = client.db("yourDatabaseName"); // Specify your database name
        const collection = database.collection("yourCollectionName"); // Specify your collection name


        // Example: Find all documents in the collection
        const documents = await collection.find({}).toArray();
        console.log("Found documents:", documents);


        // Example: Find a specific document
        const specificDocument = await collection.findOne({ name: "Example Document" });
        console.log("Specific document:", specificDocument);


    } finally {
```

```
        // Close the connection when done or if an error occurs
        await client.close();
        console.log("Connection closed.");
    }
}

connectAndOperate().catch(console.error);
```

Experiment-15:
Create an Online fee payment form using JScript and MongoDB.

To create an **Online Fee Payment Form** using **JavaScript (JScript)** and **MongoDB**, we'll follow a **Node.js + Express + MongoDB** approach — this is standard and works well for full-stack development.

✅ **Tech Stack:**

- **Frontend**: HTML + Vanilla JavaScript
- **Backend**: Node.js with Express
- **Database**: MongoDB (local or Atlas)

🔷 **Step 1: Setup MongoDB**

You can:

- Install **MongoDB locally** OR
- Use **MongoDB Atlas** (cloud-based, free tier available)

Create a collection named fee_payments in a database called college.

🔷 **Step 2: HTML + JavaScript (Frontend)**

Save as index.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Online Fee Payment Form</title>
</head>
<body>
  <h2>Fee Payment Form</h2>
  <form id="feeForm">
    <label>Name:</label><br>
    <input type="text" id="name" required><br><br>

    <label>Roll No:</label><br>
    <input type="text" id="roll" required><br><br>

    <label>Course:</label><br>
    <input type="text" id="course" required><br><br>
```

```
  <label>Amount:</label><br>
  <input type="number" id="amount" required><br><br>

  <button type="submit">Pay Now</button>
</form>

<p id="status"></p>

<script>
  document.getElementById('feeForm').addEventListener('submit', function(e) {
    e.preventDefault();

    const data = {
      name: document.getElementById('name').value,
      roll: document.getElementById('roll').value,
      course: document.getElementById('course').value,
      amount: document.getElementById('amount').value
    };

    fetch('http://localhost:3000/payfee', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(data)
    })
    .then(res => res.json())
    .then(result => {
      document.getElementById('status').innerText = result.message;
    })
    .catch(err => {
```

```
        document.getElementById('status').innerText = "Error processing payment.";
        console.error(err);
      });
    });
  </script>
</body>
</html>
```

🔷 Step 3: Node.js + Express Server (Backend)

Create a file server.js:

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();
app.use(bodyParser.json());
app.use(cors());

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/college', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

// Schema
const feeSchema = new mongoose.Schema({
  name: String,
  roll: String,
  course: String,
  amount: Number,
  paidOn: { type: Date, default: Date.now }
```

```
});

const Fee = mongoose.model('Fee', feeSchema);

// POST API to store form data
app.post('/payfee', async (req, res) => {
  try {
    const newPayment = new Fee(req.body);
    await newPayment.save();
    res.json({ message: "Fee paid successfully!" });
  } catch (err) {
    res.status(500).json({ message: "Payment failed!" });
  }
});

// Start server
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

🔷 Step 4: Run the App

1. Make sure MongoDB is running.
2. Install dependencies:

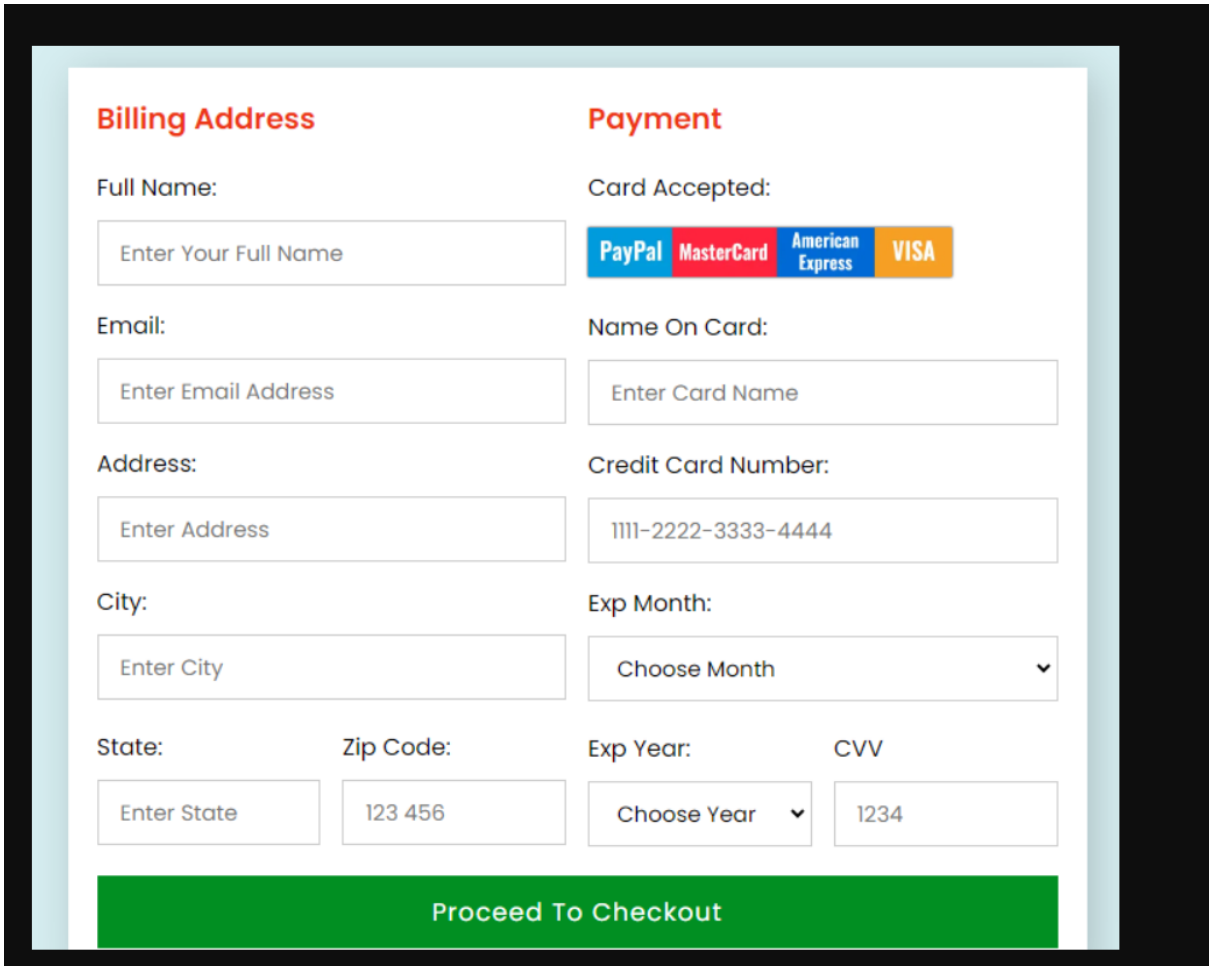
npm init -y

npm install express mongoose body-parser cors

Start server:

node server.js

Open index.html in a browser and submit the form.


Billing Address
Full Name:
Enter Your Full Name
Email:
Enter Email Address
Address:
Enter Address
City:
Enter City
State:
Enter State
Zip Code:
123 456
Payment
Card Accepted:
PayPal
MasterCard
American Express
VISA
Name On Card:
Enter Card Name
Credit Card Number:
1111-2222-3333-4444
Exp Month:
Choose Month
Exp Year:
Choose Year
CVV
1234
Proceed To Checkout